

311-CD-502-001

EOSDIS Core System Project

**Release 5A
Interoperability Subsystem (IOS)
Database Design and Database Schema
Specifications for the ECS Project**

Final

May 1999

Raytheon Systems Company
Upper Marlboro, Maryland

**Release 5A
Interoperability Subsystem (IOS)
Database Design and Database Schema
Specifications for the ECS Project**

Final

May 1999

Prepared Under Contract NAS5-60000
CDRL Item #050

RESPONSIBLE ENGINEER

Maureen Muganda /s/ 5/27/99
Maureen Muganda Date
EOSDIS Core System Project

SUBMITTED BY

Mary Armstrong /s/ 5/27/99
Mary Armstrong, Development Engineering Manager Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document describes the database design and database schema specifications for the Interoperability Subsystem (IOS). It is one of eight documents comprising the detailed database design and database schema specifications for the as-delivered ECS subsystems. A complete list of the eight documents follows:

311-CD-500 Data Management (DM) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-501 Ingest Subsystem (INS) Database Design and Database Schema Specifications for the ECS Project

311-CD-502 Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project

311-CD-503 Planning and Data Processing Subsystem (PDPS) Database Design and Database Schema Specifications for the ECS Project

311-CD-504 Science Data Server (SDSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-505 Storage Management (STMGT) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-506 Subscription Server (SUBSRV) Subsystem Database Design and Database Schema Specifications for the ECS Project

311-CD-507 Management Support Subsystem (MSS) Database Design and Database Schema Specifications for the ECS Project

This submittal meets the milestone specified in the Contract Data Requirements List (CDRL) of NASA Contract NAS5-60000. It is a formal contract deliverable with an approval code 2. As such, it does not require formal Government approval, however, the Government reserves the right to request changes within 45 days of the initial submittal. Once approved, contractor changes to this document are handled in accordance with Class I and Class II change control requirements described in the EOS Configuration Management Plan, and changes to this document shall be made by document change notice (DCN) or by complete revision.

Entity relationship diagrams (ERDs) presented in this document have been exported directly from software tools and in some cases contain too much detail to be easily readable within hard copy page constraints. The reader is encouraged to view these diagrams in Portable Document Format (PDF) on the ECS Data Handling System (EDHS) world wide web (WWW) site. The universal resource locator (URL) is: <http://edhs1.gsfc.nasa.gov>. Any questions should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Drive
Upper Marlboro, Maryland 20774-5301

Abstract

This document outlines the Release 5A “as-built” database design and database schema specifications for the Interoperability Subsystem (IOS). It includes the entity relationship diagram (ERD), physical database table definitions, and database software listings for triggers, procedures, and scripts. The ERD describes data entities and the association between these entities used within the IOS. Other information is also included to support database installation and life cycle maintenance.

Keywords: data, database, design, specifications, configuration, installation, parameters, scripts, security, data model, replication, performance tuning, SQL server, Sybase, database security, triggers, procedures, scripts.

This page intentionally left blank.

Change Information Page

List of Effective Pages			
Page Number	Issue		
Title	Submitted as Final		
iii through xiv	Submitted as Final		
1-1 through 1-2	Submitted as Final		
2-1 through 2-2	Submitted as Final		
3-1 through 3-94	Submitted as Final		
4-1 through 4-6	Submitted as Final		
5-1 through 5-4	Submitted as Final		
6-1 through 6-2	Submitted as Final		
A-1 through A-8	Submitted as Final		
AB-1 through AB-2	Submitted as Final		
Document History			
Document Number	Status/Issue	Publication Date	CCR Number
311-CD-502-001	Submitted as Final	May 1999	99-0466

This page intentionally left blank.

Contents

Preface

Abstract

1. Introduction

1.1	Identification	1-1
1.2	Scope	1-1
1.3	Purpose.....	1-1
1.4	Audience.....	1-1

2. Related Documents

2.1	Applicable Documents	2-1
2.2	Information Documents.....	2-2

3. Database Design

3.1	Design Overview	3-1
3.1.1	Physical Data Model Entity Relationship Diagram.....	3-1
3.1.2	Database Table Specifications.....	3-3
3.1.3	Column Specifications	3-23
3.1.4	Column Domains	3-45
3.1.5	Column Default Values	3-46

3.1.6	Referential Integrity Rules	3-46
3.1.7	Views.....	3-46
3.1.8	Declarative Integrity Constraints.....	3-46
3.1.9	Triggers	3-51
3.1.10	Stored Procedures.....	3-60
3.2	Flat File Usage.....	3-93
3.2.1	File Descriptions	3-93
3.2.2	Field Specifications.....	3-93
3.2.3	Domain Definitions	3-93

4. Performance and Tuning Factors

4.1	Indexes	4-1
4.2	Segments	4-1
4.3	Named Caches.....	4-2

5. Database Security

5.1	Approach	5-1
5.2	Initial Users	5-2
5.3	Groups	5-2
5.4	Roles.....	5-2
5.5	Login/Group Object Permissions	5-3

6. Scripts

6.1	Installation Scripts.....	6-1
6.2	De-Installation Scripts.....	6-1
6.3	Backup and Recovery Scripts	6-2
6.4	Miscellaneous Scripts	6-2

Figures

3-1	ERD Key	3-2
5-1	Sybase General Approach to SQL Server Security	5-1

Tables

Table 3-1.	Database Tables	3-3
Table 3-2.	EcDbVersions	3-4
Table 3-3.	IoAdAdvDescription.....	3-5
Table 3-4.	IoAdAdvMaster	3-6
Table 3-5.	IoAdAdvMasterKey.....	3-6
Table 3-6.	IoAdAdvUR	3-7
Table 3-7.	IoAdAdvXref	3-7
Table 3-8.	IoAdAlgorithmPackage	3-7
Table 3-9.	IoAdAltitudeResolution.....	3-8
Table 3-10.	IoAdAnalysisCollectionXref	3-8
Table 3-11.	IoAdAnalysisSource	3-8
Table 3-12.	IoAdApprovedAdv	3-8
Table 3-13.	IoAdBoundingRectangle.....	3-9
Table 3-14.	IoAdCampaign.....	3-9
Table 3-15.	IoAdCampaignCollectionXref	3-9
Table 3-16.	IoAdCollectionXref	3-9
Table 3-17.	IoAdContact.....	3-10
Table 3-18.	IoAdContactAddress.....	3-10
Table 3-19.	IoAdContactAdvXref.....	3-10
Table 3-20.	IoAdContactEmail	3-11
Table 3-21.	IoAdContactKey	3-11
Table 3-22.	IoAdContactOrganization	3-11
Table 3-23.	IoAdContactPerson	3-12

Table 3-24. IoAdContactTelephone	3-12
Table 3-25. IoAdDepthResolution	3-12
Table 3-26. IoAdECSCollection	3-13
Table 3-27. IoAdECSCollectionKeyword.....	3-13
Table 3-28. IoAdGroup	3-14
Table 3-29. IoAdGroupKey.....	3-14
Table 3-30. IoAdInstallableServiceAdv	3-14
Table 3-31. IoAdInstrument	3-15
Table 3-32. IoAdInstrumentCollectionXref	3-15
Table 3-33. IoAdMimeTypeAdv	3-15
Table 3-34. IoAdMultipleTypeCollection.....	3-16
Table 3-35. IoAdPersonOrganizationXref	3-16
Table 3-36. IoAdPlatform	3-16
Table 3-37. IoAdPlatformCollectionXref	3-16
Table 3-38. IoAdProductAdv	3-17
Table 3-39. IoAdProviderAdv.....	3-17
Table 3-40. IoAdRangeDateTime	3-17
Table 3-41. IoAdReferencePaper	3-17
Table 3-42. IoAdRefPaperCollectionXref.....	3-18
Table 3-43. IoAdRequest	3-18
Table 3-44. IoAdRequestKey.....	3-18
Table 3-45 . IoAdReview	3-19
Table 3-46. IoAdSensor	3-19
Table 3-47. IoAdSensorCollectionXref	3-19
Table 3-48. IoAdServiceAdv	3-20
Table 3-49. IoAdSignature	3-20
Table 3-50. IoAdSignatureServiceAdv	3-20
Table 3-51. IoAdSingleTypeCollection	3-21
Table 3-52. IoAdSiteInfo.....	3-21
Table 3-53. IoAdSpatial	3-21

Table 3-54. IoAdSpatialKeywordClass	3-22
Table 3-55. IoAdTemporalKeywordClass	3-22
Table 3-56. Dependencies on Table: IoAdAdvMaster.....	3-46
Table 3-57. Dependencies on Table: IoAdAnalysisSource	3-47
Table 3-58. Dependencies on Table: IoAdCampaign	3-47
Table 3-59. Dependencies on Table: IoAdContact	3-47
Table 3-60. Dependencies on Table: IoAdContactOrganization	3-47
Table 3-61. Dependencies on Table: IoAdContactPerson	3-48
Table 3-62. Dependencies on Table: IoAdECSCollection.....	3-48
Table 3-63. Dependencies on Table: IoAdGroup.....	3-48
Table 3-64. Dependencies on Table: IoAdInstrument.....	3-49
Table 3-65. Dependencies on Table: IoAdMultipleTypeCollection	3-49
Table 3-66. Dependencies on Table: IoAdPlatform.....	3-49
Table 3-67. Dependencies on Table: IoAdProductAdv	3-49
Table 3-68. Dependencies on Table: IoAdProviderAdv	3-49
Table 3-69. Dependencies on Table: IoAdReferencePaper.....	3-50
Table 3-70. Dependencies on Table: IoAdSensor	3-50
Table 3-71. Dependencies on Table: IoAdServiceAdv	3-50
Table 3-72. Dependencies on Table: IoAdSignatureServiceAdv	3-50
Table 3-73. Dependencies on Table: IoAdSingleTypeCollection.....	3-51
Table 3-74. Dependencies on Table: IoAdSiteInfo	3-51
Table 3-75. Dependencies on Table: IoAdSpatial.....	3-51
Table 3-76. Trigger Descriptions	3-52
Table 3-77. Summary List of Stored Procedures	3-60
Table 4-1. Index Key	4-1
Table 4-2. Indexes	4-2
Table 4-3. Segment Descriptions	4-5
Table 5-1. Permission Key	5-3
Table 5-2. Group Specifications.....	5-3
Table 6-1. Installation Scripts	6-1

Table 6-2. De-Installation Scripts	6-1
Table 6-3. Backup and Recovery Scripts	6-2
Table 6-4. Miscellaneous Scripts	6-2

Appendix A. Interoperability Subsystem Entity Relationship Diagrams

Abbreviations and Acronyms

1. Introduction

1.1 Identification

This *Interoperability Subsystem (IOS) Database Design and Database Schema Specifications* document, Contract Data Requirement List (CDRL) Item Number 050, whose requirements are specified in Data Item Description (DID) 311/DV2, is a required deliverable under the Earth Observing System (EOS) Data and Information System (EOSDIS) Core System (ECS), Contract NAS5-60000.

1.2 Scope

The *IOS Database Design and Database Schema Specifications* document describes the database that supports the data requirements for the IOS, Release 5A .

1.3 Purpose

The purpose of the *IOS Database Design and Database Schema Specifications* document is to support the administrators of the IOS database throughout its life cycle. Additionally, this document communicates the database specifications in sufficient detail to support other ongoing installation and operational activities (e.g., configuration management, data administration, system installation and maintenance).

1.4 Audience

The IOS *Database Design and Database Schema Specifications* document is intended to be used and maintained by the ECS maintenance and operations staff. The document is organized as follows:

Section 1 provides information regarding the identification, purpose, scope, and audience.

Section 2 provides a listing of related documents used to develop this document.

Section 3 contains an overview of the database design including the entity relationship diagram (ERD) representing the physical data model, the database tables and columns, flat files and fields, triggers, and stored procedures.

Section 4 provides a description of database performance and tuning features, (i.e., indexes, caches, and data segments) for the IOS database implementation.

Section 5 provides a high level description of the preliminary security infrastructure including listings of anticipated users, groups, and permissions expected for preliminary operational use.

Section 6 provides listings of the scripts used for database installation, de-installation, backup and recovery, and other miscellaneous administration functions.

This page is left intentionally blank.

2. Related Documents

2.1 Applicable Documents

The following documents, including Internet links, are referenced in this document, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this volume.

305-CD-500	Release 5A Segment Design Specification for the ECS Project
920-TDG-009	DAAC Hardware Database Mapping/GSFC
920-TDN- 009	DAAC Hardware Database Mapping/NSIDC
920-TDE-009	DAAC Hardware Database Mapping/EDC
920-TDL-009	DAAC Hardware Database Mapping/LARC
920-TDS-009	DAAC Hardware Database Mapping/SMC
920-TDG-010	DAAC Database Configuration/GSFC
920-TDN-010	DAAC Database Configuration/NSIDC
920-TDE-010	DAAC Database Configuration/EDC
920-TDL-010	DAAC Database Configuration/LARC
920-TDS-010	DAAC Database Configuration/SMC
920-TDG-011	DAAC Sybase Log Mapping/GSFC
920-TDN-011	DAAC Sybase Log Mapping/NSIDC
920-TDE-011	DAAC Sybase Log Mapping/EDC
920-TDL-011	DAAC Sybase Log Mapping/LARC
920-TDS-011	DAAC Sybase Log Mapping/SMC
922-TDG-013	Disk Partitions/GSFC
922-TDN-013	Disk Partitions/NSIDC
922-TDE-013	Disk Partitions/EDC
922-TDL-013	Disk Partitions/LARC
922-TDS-013	Disk Partitions/SMC

These documents are maintained as part of the ECS baseline and available on the world wide web at the URL: <http://cmdm.east.hitc.com/baseline>. Please note that this is a partial mirror site in that some items are not available (they are identified) since this is OPEN to all. This site may also be reached through the EDHS homepage. Scroll page to the connections line and click on the ECS Baseline Information System link.

2.2 Information Documents

The following documents, although not directly applicable, amplify or clarify the information presented in this document. These documents are not binding on this document.

313-CD-500	Release 5A CSMS/SDPS Internal ICD for the ECS Project
609-CD-500	Release 5A Operations Tools Manual for the ECS Project
611-CD-500	Release 5A Mission Operation Procedures for the ECS Project

These documents are accessible via the EDHS homepage.

3. Database Design

3.1 Design Overview

Data requirements for IOS fall into four logical groupings:

Advertising information – data describing the approved or pending

Collection metadata – data summarizing the science content of advertised ECS data collections.

Primary key management - data used to generate a unique primary key for a table in the IOS database.

Database initialization data – data used to initially populate tables in the IOS database on a clean install.

The IOS database implements a majority of the persistent data requirements for the IOS database. Other non-database data requirements, used for system support, are implemented in flat files—see Section 3.2 for descriptions of these flat files. The database is designed to satisfy IOS business rules while maintaining data integrity, consistency, and performance. Database tables are implemented using the Sybase Relational Database Management System (RDBMS). All components of the IOS database and flat files are described in the following sections; information is presented in sufficient detail to support operational needs.

3.1.1 Physical Data Model Entity Relationship Diagram

Entity relationship diagrams (ERDs) were developed for use as a “roadmap” to the IOS database. These ERDs are found in Appendix A. An ERD is a schematic of the physical data structure that illustrates the dependencies and relationships between database entities (i.e., tables). On ERDs, database entities are represented by rectangles and relationships are represented by arrows as shown by the key in Figure 3-1. Details on the syntax used by the *S-Designor Data Architect* Computer Aided Software Engineering (CASE) tool may be found in the *Powersoft: S-Designor for PowerBuilder* Reference Guide. The ERDs presented in Appendix A for the IOS database were produced using the *S-Designor* tool.

The ECS Conceptual Model for the Science Data Processing Segment (SDPS) was developed using an Object Oriented (OO) CASE tool. However, since Sybase implements a RDBMS with an Object wrapper, the syntax (model notation) is converted from OO to relational and the terminology changes—the “attribute” becomes “column” and “class” becomes “table.” Since the specifications of some entities in this document are transferred from the OO Conceptual Model repository, there are many cases where the OO terminology is retained—as, for example, in the table and column names and definitions.

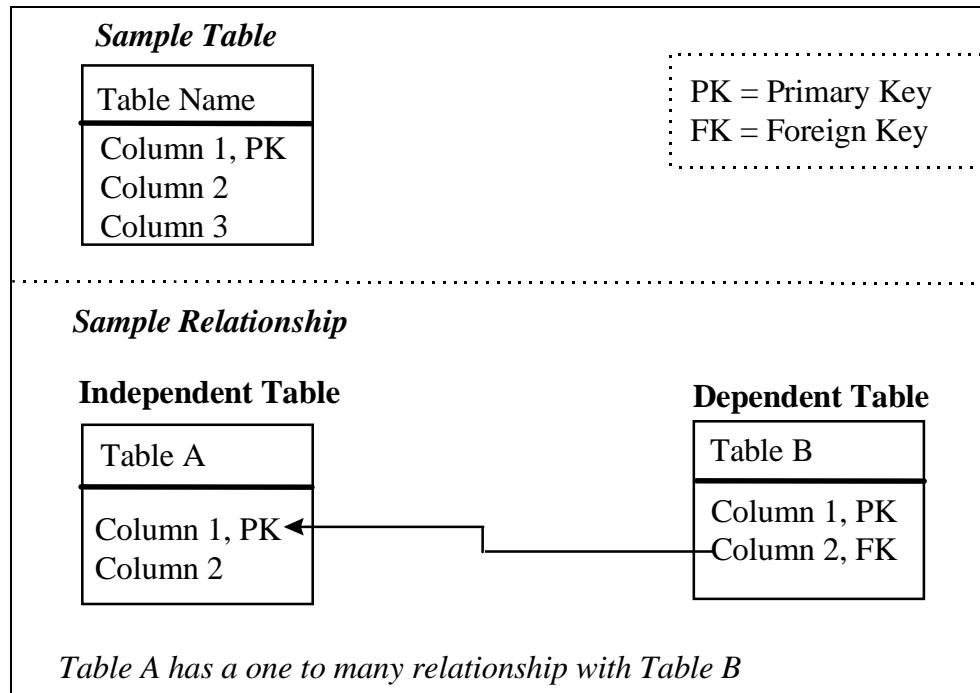


Figure 3-1. ERD Key

3.1.2 Database Table Specifications

Table 3-1 contains a listing of all the database tables named within the IOS database. This list is presented in alphabetical order corresponding to the database tables illustrated in the ERD (reference Figure 3-2). The database tables listed immediately following Table 3-1 are presented in the same order as the table.

Table 3-1. Database Tables (1 of 2)

Database Table Name	Logical Grouping
IoAdAdvMaster	Advertisement Information
IoAdAdvMasterKey	Primary Key Management
IoAdAdvUR	Advertisement Information
IoAdAdvXref	Advertisement Information
IoAdAlgorithmPackage	Collection Metadata
IoAdAltitudeResolution	Collection Metadata
IoAdAnalysisCollectionXref	Collection Metadata
IoAdAnalysisSource	Collection Metadata
IoAdApprovedAdv	Advertisement Information
IoAdBoundingRectangle	Collection Metadata
IoAdCampaign	Collection Metadata
IoAdCampaignCollectionXref	Collection Metadata
IoAdCollectionXref	Collection Metadata
IoAdContact	Collection Metadata
IoAdContactAddress	Collection Metadata
IoAdContactAdvXref	Collection Metadata
IoAdContactEmail	Collection Metadata
IoAdContactKey	Primary Key Management
IoAdContactOrganization	Collection Metadata
IoAdContactPerson	Collection Metadata
IoAdContactTelephone	Collection Metadata
IoAdDepthResolution	Collection Metadata
IoAdECSCollection	Collection Metadata
IoAdECSCollectionKeyword	Collection Metadata
IoAdGroup	Advertisement Information
IoAdGroupKey	Primary Key Management
IoAdInstallableServiceAdv	Advertisement Information
IoAdInstrument	Collection Metadata
IoAdInstrumentCollectionXref	Collection Metadata
IoAdMimeTypeAdv	Advertisement Information
IoAdMultipleTypeCollection	Collection Metadata
IoAdPersonOrganizationXref	Collection Metadata
IoAdPlatform	Collection Metadata
IoAdPlatformCollectionXref	Collection Metadata

Table 3-1. Database Tables (2 of 2)

Database Table Name	Logical Grouping
IoAdProductAdv	Advertisement Information
IoAdProviderAdv	Advertisement Information
IoAdRangeDateTime	Collection Metadata
IoAdReferencePaper	Collection Metadata
IoAdRefPaperCollectionXref	Collection Metadata
IoAdRequest	Advertisement Information
IoAdRequestKey	
IoAdReview	Collection Metadata
IoAdSensor	Collection Metadata
IoAdSensorCollectionXref	Collection Metadata
IoAdServiceAdv	Advertisement Information
IoAdSignature	Advertisement Information
IoAdSignatureServiceAdv	Advertisement Information
IoAdSingleTypeCollection	Collection Metadata
IoAdSiteInfo	Advertisement Information
IoAdSpatial	Collection Metadata
IoAdSpatialKeywordClass	Collection Metadata
IoAdTemporalKeywordClass	Collection Metadata

The following report is produced by the S-Designor CASE tool and edited for format consistency. The report provides specifications on the IOS database tables. The report is sorted in alphabetical order by table name. Specifications include the table name, a brief description of the table, and the columns comprising the table. The column information includes the column name and the column attributes, i.e., type (format of the data stored within the database), primary key indicator(s), and a mandatory indicator for determining if the column must contain data when the row exists. In some cases the content of the column specification “Type” will reference a domain value (refer to Section 3.1.4 for more information on the domain values).

Table 3-2. EcDbVersions (1 of 2)

Description: This identifies the current version level of the IOS database

Column List

Name	Type	P	M
EcDbSchemaVersionID	Smallint	Yes	Yes
EcDbDropVersion	Char(64)	No	Yes
EcDbDropDescription	Varchar(255)	No	Yes

Table 3-2. EcDbVersions (2 of 2)

Name	Type	P	M
EcDbCCurrentVersionFlag	Char(1)	No	Yes
EcDbDatabaseName	Varchar(255)	No	No
EcDbDropInstallDate	Datetime	No	No
EcDbSybaseVersion	Varchar(255)	No	No
EcDbSybaseServer	Varchar(255)	No	No
EcDbComments	Varchar(255)	No	No
EcDbUpdateProcess	Varchar(255)	No	No

Table 3-3. IoAdAdvDescription

Description: This table stores a general description of an advertisement.

Column List

Name	Type	P	M
advId	Numeric(8)	Yes	Yes
description	Varchar(255)	No	Yes
seqNumber	Int	Yes	Yes

Table 3-4. IoAdAdvMaster

Description: This table stores the general information of approved advertisements.

Column List:

Name	Type	P	M
AdvGroupId	numeric(8)	No	Yes
AdvId	numeric(8)	Yes	Yes
AdvType	varchar(80)	No	Yes
ExpirationDate	datetime	No	No
LogicalServerName	varchar(40)	No	No
ServerType	varchar(40)	No	No
SiteId	int	No	Yes
StartDate	datetime	No	Yes
Title	varchar(100)	No	Yes
UniqueId	numeric(6)	No	Yes
UpperTitle	varchar(100)	No	No

Table 3-5. IoAdAdvMasterKey

Description: This is a look up table to maintain the highest value of advId.

Column List:

Name	Type	P	M
advIdUnique	numeric(6)	Yes	Yes

Table 3-6. IoAdAdvUR

Description: This table stores the address of the service advertisement universal reference.

Column List:

Name	Type	P	M
advUR	varchar(255)	No	Yes
seqNumber	int	Yes	Yes
serviceId	numeric(8)	Yes	Yes

Table 3-7. IoAdAdvXref

Description: This table maintains the references between product advertisement and service advertisement.

Column List

Name	Type	P	M
leftAdvId	numeric(8)	Yes	Yes
rightAdvId	numeric(8)	Yes	Yes

Table 3-8. IoAdAlgorithmPackage

Description: This table stores the algorithm package information associated with the single type collections.

Column List:

Name	Type	P	M
AlgorithmPackageVersion	char(80)	Yes	Yes
AlgorithmPackageName	char(80)	Yes	Yes
collectionId	numeric(8)	No	No
contactId	numeric(8)	No	No
PackageAcceptanceDate	datetime	No	No

Table 3-9. *IoAdAltitudeResolution*

Description: This table stores the altitude resolution associated with the ECS collections.

Column List:

Name	Type	P	M
AltitudeResolution	float	No	Yes
collectionId	numeric(8)	Yes	Yes

Table 3-10. *IoAdAnalysisCollectionXref*

Description: This table maintains the cross-reference between analysis sources and the ECS collections.

Column List:

Name	Type	P	M
AnalysisShortName	char(20)	Yes	Yes
collectionId	numeric(8)	Yes	Yes

Table 3-11. *IoAdAnalysisSource*

Description: This table defines analysis sources germane to ECS.

Column List:

Name	Type	P	M
AnalysisLongName	varchar(80)	No	No
AnalysisShortName	char(20)	Yes	Yes

Table 3-12. *IoAdApprovedAdv*

Description: This table identifies approved advertisements.

Column List:

Name	Type	P	M
advId	numeric(8)	Yes	Yes

Table 3-13. IoAdBoundingRectangle

Description: This table stores the spatial area coverage for ECS collection.

Column List:

Name	Type	P	M
CollectionId	numeric(8)	Yes	Yes
EastBoundingCoordinate	numeric(11,6)	No	Yes
NorthBoundingCoordinate	numeric(11,6)	No	Yes
SouthBoundingCoordinate	numeric(11,6)	No	Yes
WestBoundingCoordinate	numeric(11,6)	No	Yes

Table 3-14. IoAdCampaign

Description: This table stores the scientific endeavors to which the collection is associated.

Column List:

Name	Type	P	M
CampaignLongName	varchar(80)	No	No
CampaignShortName	char(20)	Yes	Yes

Table 3-15. IoAdCampaignCollectionXref

Description: This table maintains the cross-reference between campaign and ECS collection.

Column List:

Name	Type	P	M
CampaignShortName	char(20)	Yes	Yes
CollectionId	numeric(8)	Yes	Yes

Table 3-16. IoAdCollectionXref

Description: This table contains the cross-reference between single type collection and multiple type collection.

Column List:

Name	Type	P	M
collectionIdAgg	numeric(8)	Yes	Yes
collectionIdRef	numeric(8)	Yes	Yes

Table 3-17. IoAdContact

Description: This table stores the contact information of the advertisement.

Column List:

Name	Type	P	M
contactId	numeric(8)	Yes	Yes
contactRole	char(16)	No	Yes
siteId	int	No	Yes
uniqueId	numeric(6)	No	Yes

Table 3-18. IoAdContactAddress

Description: This table stores the address of points of contact for the advertisement.

Column List:

Name	Type	P	M
addressId	numeric(8)	Yes	Yes
City	char(30)	No	No
contactId	numeric(8)	Yes	Yes
Country	char(10)	No	No
PostalCode	char(20)	No	No
StateProvince	char(20)	No	No
StreetAddress	char(80)	No	No

Table 3-19. IoAdContactAdvXref

Description: This table contains the cross-reference between contact and advertisement.

Column List:

Name	Type	P	M
advId	numeric(8)	Yes	Yes
contactId	numeric(8)	Yes	Yes

Table 3-20. IoAdContactEmail

Description: This table stores the Email address of the points of contact for the advertisement.

Column List:

Name	Type	P	M
contactId	numeric(8)	No	No
ElectronicEmailAddress	varchar(255)	Yes	Yes

Table 3-21. IoAdContactKey

Description: This is a look up table to maintain the highest value of the contactId.

Column List:

Name	Type	P	M
contactIdUnique	numeric(6)	Yes	Yes

Table 3-22. IoAdContactOrganization

Description: This table stores the name of the organizations.

Column List:

Name	Type	P	M
ContactOrganizationName	varchar(255)	No	No
organizationId	numeric(8)	Yes	Yes

Table 3-23. IoAdContactPerson

Description: This table stores the name of the points of contact for the advertisement.

Column List:

Name	Type	P	M
ContactFirstName	char(30)	No	Yes
contactId	numeric(8)	Yes	Yes
ContactJobPosition	varchar(255)	No	No
ContactLastName	varchar(30)	No	Yes
ContactMiddleName	varchar(30)	No	No

Table 3.24. IoAdContactTelephone

Description: This table stores the phone information of the points of contact for the advertisement.

Column List:

Name	Type	P	M
contactId	numeric(8)	No	No
TelephoneNumber	char(23)	Yes	Yes
TelephoneNumberType	char(5)	No	Yes

Table 3-25. IoAdDepthResolution

Description: This table stores the depth resolution associated with ECS collections.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
DepthResolution	float	No	Yes

Table 3-26. IoAdECSCollection

Description: This table stores the ECS collection associated to the product advertisement.

Column List:

Name	Type	P	M
ArchiveCenter	char(20)	No	Yes
collectionId	numeric(8)	Yes	Yes
guideURL	varchar(100)	No	No
ProcessingCenter	char(20)	No	No
productURL	varchar(100)	No	No
RevisionDate	datetime	No	No
ShortName	char(8)	No	Yes
SuggestedUsage	text	No	No
VersionID	int	No	Yes
VersionDescription	varchar(255)	No	No

Table 3-27. IoAdECSCollectionKeyword

Description: This table stores all types of keywords used in searching ECS collections.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
ECSDisciplineKeyword	char(24)	No	No
ECSParameterKeyword	char(80)	No	No
ECSTermKeyword	char(50)	No	No
ECSTopicKeyword	char(32)	No	No
ECSVariableKeyword	char(80)	No	No
keywordId	numeric(8)	Yes	Yes

Table 3-28. IoAdGroup

Description: This table stores the moderation group for the advertisement.

Column List:

Name	Type	P	M
acl	char(50)	No	No
groupDescription	varchar(255)	No	No
groupId	numeric(8)	Yes	Yes
groupName	char(50)	No	No
moderator	char(50)	No	No
siteld	int	No	Yes
uniqueId	numeric(6)	No	Yes

Table 3-29. IoAdGroupKey

Description: This is a look up table to maintain the highest value of the groupId currently being used.

Column List:

Name	Type	P	M
groupIdUnique	numeric(6)	Yes	Yes

Table 3-30. IoAdInstallableServiceAdv

Description: This table stores the software information which is installed to the client site upon request.

Column List:

Name	Type	P	M
ftpURL	varchar(100)	No	No
packageSize	int	No	No
serviceId	numeric(8)	Yes	Yes

Table 3-31. *IoAdInstrument*

Description: This table stores the instrument information associated with ECS collection.

Column List:

Name	Type	P	M
InstrumentLongName	char(80)	No	No
InstrumentShortName	char(20)	Yes	Yes

Table 3-32. *IoAdInstrumentCollectionXref*

Description: This table maintains the cross-reference between ECS collection and Instrument.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
InstrumentShortName	char(20)	Yes	Yes

Table 3-33. *IoAdMimeServiceAdv*

Description: This table stores the MIME (Multipurpose Internet Mail Extension) service advertisement.

Column List:

Name	Type	P	M
serviceId	numeric(8)	Yes	Yes
serviceURL	varchar(100)	No	No

Table 3-34. *IoAdMultipleTypeCollection*

Description: This table stores the multiple type ECS collections.

Column List:

Name	Type	P	M
AggregationType	char(20)	No	No
collectionId	numeric(8)	Yes	Yes

Table 3-35. *IoAdPersonOrganizationXref*

Description: This table maintains the cross-reference between Contact Person and Contact Organization.

Column List:

Name	Type	P	M
ContactId	numeric(8)	Yes	Yes
OrganizationId	numeric(8)	Yes	Yes

Table 3-36. *IoAdPlatform*

Description: This table stores the platforms associated with the acquisition of the collection.

Column List:

Name	Type	P	M
PlatformLongName	char(80)	No	No
PlatformShortName	char(20)	Yes	Yes

Table 3-37. *IoAdPlatformCollectionXref*

Description: This table maintains the cross-reference between Platform and ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
PlatformShortName	char(20)	Yes	Yes

Table 3-38. IoAdProductAdv

Description: This table stores the product advertisement.

Column List:

Name	Type	P	M
productId	numeric(8)	Yes	Yes
providerId	numeric(8)	No	No

Table 3-39. IoAdProviderAdv

Description: This table stores the provider advertisement.

Column List:

Name	Type	P	M
providerId	numeric(8)	Yes	Yes
providerURL	varchar(100)	No	No

Table 3-40. IoAdRangeDateTime

Description: This table stores the start/end time of a collection.

Column List:

Name	Type	P	M
CollectionId	numeric(8)	Yes	Yes
PresentFlag	numeric(1)	No	No
RangeBeginningDate	datetime	Yes	Yes
RangeEndingDate	datetime	Yes	Yes

Table 3-41. IoAdReferencePaper

Description: This table stores the reference document of a collection.

Column List:

Name	Type	P	M
AccessInstructions	varchar(255)	No	No
AuthorAffiliation	varchar(64)	No	No
AuthorName	varchar(64)	No	No
PublicationDate	datetime	No	No
ReferencePaperReference	char(20)	Yes	Yes
Title	varchar(100)	No	No

Table 3-42. IoAdRefPaperCollectionXref

Description: This table maintains the cross-reference between Reference Paper and ECS collection.

Column List:

Name	Type	P	M
CollectionId	numeric(8)	Yes	Yes
ReferencePaperReference	char(20)	Yes	Yes

Table 3-43. IoAdRequest

Description: This table stores the information that external providers make a request to change the current approved advertisement.

Column List:

Name	Type	P	M
newAdvId	numeric(8)	Yes	Yes
oldAdvId	numeric(8)	No	No
oldAdvModTime	datetime	No	No
requestId	numeric(8)	No	No
requestTime	datetime	No	No
siteId	int	No	Yes
type	int	No	Yes
uniqueId	numeric(6)	No	Yes
who	varchar(255)	No	No
why	varchar(255)	No	No

Table 3-44. IoAdRequestKey

Description: This is a look up table to maintain the highest value of the new AdvId.

Column List:

Name	Type	P	M
requestIdUnique	numeric(6)	Yes	Yes

Table 3-45. IoAdReview

Description: This table stores the dates for the QA peer review for a single type collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
FutureReviewDate	datetime	No	No
reviewId	numeric(8)	Yes	Yes
ScienceReviewDate	datetime	No	No

Table 3-46. IoAdSensor

Description: This table stores the sensory subcomponents of an instrument.

Column List:

Name	Type	P	M
SensorLongName	char(80)	No	No
SensorShortName	char(20)	Yes	Yes

Table 3-47. IoAdSensorCollectionXref

Description: This table maintains the cross-reference between Sensors and ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
SensorShortName	char(20)	Yes	Yes

Table 3-48. IoAdServiceAdv

Description: This table stores the service advertisement.

Column List:

Name	Type	P	M
providerId	numeric(8)	No	No
serviceId	numeric(8)	Yes	Yes

Table 3-49. IoAdSignature

Description: This table stores the signature service associated with the signature service advertisement.

Column List:

Name	Type	P	M
GIParamList	varchar(255)	No	Yes
seqNumber	int	Yes	Yes
serviceId	numeric(8)	Yes	Yes

Table 3-50. IoAdSignatureServiceAdv

Description: This table stores the signature service advertisement. Signature service is the C++ operation to perform the service.

Column List:

Name	Type	P	M
internalName	varchar(100)	No	Yes
serviceClass	varchar(100)	No	Yes
serviceId	numeric(8)	Yes	Yes
serviceName	varchar(100)	No	Yes
serviceSignatureId	numeric(8)	No	No
serviceURId	numeric(8)	No	No

Table 3-51. *IoAdSingleTypeCollection*

Description: This table stores the information specific to a single type collection.

Column List:

Name	Type	P	M
AccessConstraints	varchar(255)	No	No
CitationforExternalPublication	varchar(255)	No	No
collectionId	numeric(8)	Yes	Yes
MaintenanceandUpdateFrequency	varchar(80)	No	No
ProcessingLevelID	char(32)	No	No

Table 3-52. *IoAdSiteInfo*

Description: This table stores the DAAC which an advertisement belongs to.

Column List:

Name	Type	P	M
local	bit	No	Yes
sited	int	Yes	Yes
siteName	varchar(40)	No	Yes

Table 3-53. *IoAdSpatial*

Description: This table stores the spatial extent of ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
LatitudeResolution	float	No	No
LongitudeResolution	float	No	No
SpatialCoverageType	char(25)	No	Yes

Table 3-54. *IoAdSpatialKeywordClass*

Description: This table stores the keyword of spatial region for ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
SpatialKeyword	char(10)	Yes	Yes

Table 3-55. *IoAdTemporalKeywordClass*

Description: This table stores the keyword of temporal characteristics for ECS collection.

Column List:

Name	Type	P	M
collectionId	numeric(8)	Yes	Yes
TemporalKeyword	char(10)	Yes	Yes

3.1.3 Column Specifications

Brief definitions of each of the columns¹ within the IOS database and their applicable valid values, or references to other documents containing the valid values, are contained herein.

Column: AccessConstraints

Description: Restrictions and legal prerequisites for accessing the collection. These include any access constraints applied to assure the protection of privacy or intellectual property, and any special restrictions or limitations on obtaining the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AccessInstructions

Description: Instructions describing how to obtain electronic access to a stand-alone document. May simply be an anonymous ftp site address, or a World Wide Web homepage URL.

Column: acl

Description: The name of the access control list used by the moderator to create/update/delete approved advertisement for the given group.

Column: addressId

Description: The unique ID which identifies the address of the contact.

Column: advGroupId

Description: The IDs which uniquely identify the group that advertisement belong to.

¹ The term "column" is used interchangably with the term "attribute" where it refers to a database structure. The ECS Conceptual Model was developed using Object Oriented terminology. This model is converted to a relational structure for physical implementation within Sybase. The descriptions of some "columns" are derived from the Object Oriented Conceptual Model.

Column: advId

Description: An ID which uniquely identifies an advertisement.

Note: AdvId is generated from siteId and uniqueId. Refer to stored procedure "IoAdgetAdvMasterKey".

Column: advIdUnique

Description: The last value of advId used.

Note: This key is used by the store procedure "IoAdAdvMasterKey" to generate the next advId in sequence.

Column: advType

Description: The type of advertisement.

Column: advUR

Description: The address of the service advertisement UR.

Column: AggregationType

Description: This attribute will contain the criteria by which multiple type collections have been grouped. It will describe the major categorization which applies to the data therein. Possible collection groupings include: INSTRUMENT, for all collections associated with a given collecting instrument such as CERES-this is a common aggregation criteria for ECS 'dataset'; PROJECT, for all data associated with a given project that may or may not be related to a single instrument, such as FIRE-this is often an aggregation criteria for ECS 'products'; SUPERGRANULE, for collections of granules that a data provider wishes to be orderable as a single related grouping, such as SSM/I TIME SERIES-this is a concept adopted from MSFC use; EVENT, for a predetermined/tagged set of granules that have been found to be related to a particular geophysical phenomena or event, such as MIDWEST FLOOD '93 or OZONE HOLE or MT.PINATUBO-this is a new ECS concept, also suggested by the University of Virginia Atmospheric researchers. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AlgorithmPackageVersion

Description: This attribute specifies the version of the full package being delivered. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AlgorithmPackageName

Description: This attribute is the name given to the complete delivered package submitted for algorithm integration and test. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AltitudeResolution

Description: The minimum distance possible between two adjacent altitude values, expressed in Altitude Distance Units of measure. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015.

Column: AnalysisLongName

Description: The expanded or long name of the analysis source identified using AnalysisShortName. AnalysisLongName is intended to categorize collections by the processes which collected (e.g., census survey) or produced them (e.g., NMC 16-level Nested Grid Model). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AnalysisShortName

Description: AnalysisShortname is the unique identifier of the collection or analysis process(s) which best characterize the ECS Collection or Granule. ECS Collections or Granules may be characterized by both a collection and an analysis data set which include data collected using the NWS ASOS network (PlatformType = Network, PlatformShortName = ASOS) which was processed using an NMC analysis model (e.g., AnalysisType = Model, AnalysisShortName = RAFS, AnalysisDescription = Regional Area Forecast System, AnalysisTechnique = Regional Optimal Interpolation.) This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ArchiveCenter

Description: Center where collection is archived. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: AuthorAffiliation

Description: The name of an agency or center with which the author of the document works for or is affiliated with.

Column: AuthorName

Description: The name of the author of the reference paper.

Column: CampaignLongName

Description: The expanded name of the campaign/experiment (e.g., Global Climate Observing System). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: CampaignShortName

Description: The unique identifier by which a campaign/project/experiment is known. The campaign/project is the scientific endeavor associated with the acquisition of the collection. Collections may be associated with multiple campaigns. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: CitationforExternalPublication

Description: The recommended reference to be used when referring to this collection in publications. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: City

Description: The city of the person or organization of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: collectionId

Description: The unique ID which identifies an ECS collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: collectionIdAgg

Description: The aggregate type of collections.

Column: collectionIdRef

Description: The collection which is included in the collectionIdAgg.

Column: ContactFirstName

Description: First name of the individual to which the contact role (producer, archiver, distributor, or data originator) applies. People are points of contact, rather than organization, in cases where the association of the person to the data set is more significant than the association of the organization to the data set. They may also be included if both a single person and organization are provided as points of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: contactId

Description: The unique ID which identifies the contact information.

Note: contactId is generated from uniqueId and siteId. Refer to the store procedure "IoAdGetContactKey". This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: contactIdUnique

Description: The highest value of the ContactId that is currently used.

Column: ContactJobPosition

Description: The title of the individual (i.e., Team Leader, Principal Investigator). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ContactLastName

Description: Last name of the individual with the contact role (producer, archiver, distributor, or data originator) applies. People are points of contact, rather than organization, in cases where the association of the person to the data set is more significant than the association of the organization to the data set. They may also be included if both a single person and organization are provided as points of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ContactMiddleName

Description: The middle name of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ContactOrganizationName

Description: This is the name of the organization. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: contactRole

Description: Classification of individuals who are associated with a given collection. This is an SDSRV Subsystem column “ContactRole.”

Valid Values: See 420-TP-015

Column: Country

Description: The country of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: DepthResolution

Description: The minimum distance possible between two adjacent depth values, expressed in depth distance units of measure - mandatory. This is an SDSRV Subsystem column.

Valid Values: >0.0, See 420-TP-015

Column: EastBoundingCoordinate

Description: Eastern-most limit of coverage expressed in geodetic longitude.

Valid Values: [-180.0 <= East Bounding Coordinate <= 180.0]

Column: EcDbComment

Description: Notes or comments on the database version level.

Column: EcDbCurrentVersionFlag

Description: Flag indicating if this row represents the current database version entry

Valid Values: 1= yes, 0 = no

Column: EcDbDatabaseName

Description: The name of the database for which this database version level is applied.

Column: EcDbDropDescription

Description: The official description of the ECS software drop appropriate for this database version level.

Column: EcDbDropInstallDate

Description: The date and time that the database version level was installed.

Column: EcDbDropVersion

Description: The official name of the ECS software drop appropriate for this database version level.

Column: EcDbSchemaVersionId

Description: The subsystem-specific identifier for this database schema version.

Column: EcDbSybaseServer

Description: The name of the baseline Sybase SQL server controlling this database.

Valid Values: See 920-TDx-009

Column: EcDbSybaseVersion

Description: The software release version of the Sybase SQL server in place when this database version level was initially installed.

Column: EcDbUpdateProcess

Description: The installation method by which this database version level was installed.

Column: ECSDisciplineKeyword

Description: Keyword used to describe the general discipline area of the collection. A collection can conceivably cover several disciplines. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSParameterKeyword

Description: Keyword used to describe specific characteristics of a collection at a higher level of detail than provided by ECSVariableKeyword. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSTermKeyword

Description: Keyword used to describe the science parameter area of the collection. A collection can conceivably cover many such parameters. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSTopicKeyword

Description: Keyword used to describe the general topic area of collection. A collection can conceivably cover several topics. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ECSVariableKeyword

Description: Keyword used to describe the specific science parameter content of the collection. A collection can conceivably cover many specific parameters. The keyword valid values are the lowest level physical parameter terms which are normally searched by a user; i.e., a user enters a keyword which when found may connect with one or more parameters from collections. The

keywords are also the lowest level words which describe product content without being the server specific measurement (held in Parameter class). While there is a controlled list of these parameters, additional can be made by an as yet unspecified configuration control process. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column:ElectronicEmailAddress

Description: The address of the electronic mailbox of the organization or individual. The address, following NASA Global Change Master Directory format, should be of the form 'network name>network address'. Examples of network names are NSN, SPAN, telemail, ARPANET, and Internet. Examples of network address are NSSDCA::NG.MIKEMARTIN/NASA, MMARTIN@JPL.MILVAX, or miken@eos.hitc.com. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: expirationDate

Description: Date that the advertisement will be expired.

Column: ftpURL

Description: The address of FTP site from where the software can be downloaded.

Column: FutureReviewDate

Description: Date of next planned AQ peer review. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: GIParamList

Description: The compressed list contains metadata information.

Column: groupDescription

Description: The brief description of the group.

Column: groupId

Description: The unique IDs which identify the group.

Note: GroupId is generated from uniqueId and siteId. Refer to the stored procedure "IoAdGetGroupKey".

Column: groupIdUnique

Description: The highest value of the groupId.

Column: groupName

Description: This attribute is used to tie a “kind” of metadata to physical processing in the DBWrappers. For example: “QAGranule” relates the DsMdQaGranules table for a function (Physical Delete) and sequence of SQL code (EXEC DsDbProcDeleteQaGranule) to process the data. This is an SDSRV Subsystem column “GroupName.”

Valid Values: See 420-TP-015

Column: guideURL

Description: The address of UR where the guide information for the collection is stored.

Note: The data origin of this attribute has not been determined. Potential source is Document Data Server.

Column: InstrumentLongName

Description: The expanded name of the primary sensory instrument (e.g., Advanced Spaceborne Thermal Emission and Reflective Radiometer). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: InstrumentShortName

Description: The unique identifier of an instrument. (e.g., ASTER). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: internalName

Description: The type of service for an instrument.

Column: keywordId

Description: The unique IDs which identify the keyword.

Column: LatitudeResolution

Description: The minimum difference between two adjacent latitude values expressed in Geographic Coordinate Units of measure. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: leftAdvId

Description: The advId which represents a product advertisement.

Column: local

Description: The Boolean value to identify the local DAAC.

Column: logicalServerName

Description: The logical server name is the cross-reference name to the physical server where the advertisement data is located. It is a static identification to the physical server which may vary.

Valid Values: “sdsrv@gfsc”, “sdsrv@LaRc”, “sdsrv@edc”, “sbsrv@nsidc”, “sbsrv@gfsc”, “sbsrv@LaRc”, “sbsrv@edc”, “dim@nsidc”, “dim@gfsc”, “dim@LaRc”, “dim@edc”, “dim@nsidc”

Column: LongitudeResolution

Description: The minimum difference between two adjacent longitude values expressed in Geographic Coordinate Units of Measure. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: MaintenanceandUpdateFrequency

Description: The frequency with which changes and additions are made to the collection after the initial dataset begins to be collected/processed. This is an SDSRV Subsystem column “MaintenanceUpdateFrequency.”

Valid Values: See 420-TP-015

Column: moderator

Description: The name of the moderator assigned to the group.

Column: newAdvId

Description: The unique ID which identifies the new approved advertisement specified in additions or modifications.

Valid Values: See AdvId valid values.

Column: NorthBoundingCoordinate

Description: Northern-most coordinate of the limit of coverage expressed in geodetic latitude.

Valid Values: [-90.0 <= North Bounding Coordinate <= 90.0], [North Bounding Coordinate => South Bounding Coordinate]

Column: oldAdvId

Description: The old approved advertisement specified in deletion or modification.

Valid Values: See AdvId valid values.

Column: oldAdvModTime

Description: The modification time of the old approved advertisement.

Column: organizationId

Description: The unique ID which identifies the contact organization. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PackageAcceptanceDate

Description: The date that the particular version of the package is accepted.

Column: packageSize

Description: The size of the software in kilobytes.

Column: PlatformLongName

Description: The expanded or long name of the platform associated with an instrument. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PlatformShortName

Description: The unique identifier of a specific platform (e.g., AM1, TRMM, PM). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: PostalCode

Description: The zip or other postal code of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: presentFlag

Description: If this is 1, then it means the RangeEndingDate is present in the table, IoAdRangeDateTime. If this is 0, then the effective RangeEndingDate is the current date.

Column: ProcessingCenter

Description: Center where collection was or is being processed (i.e., name of DAAC or SCF). This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: ProcessingLevelID

Description: This attribute reflects the classification of the science data processing level, which defines in general terms the characteristics of the output of the processing performed. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: productId

Description: The unique ID which identifies the product advertisement.

Note: The productId is equivalent to AdvId in the IoAdAdvMaster table.

Valid Values: See AdvId valid values.

Column: productURL

Description: The address of UR where providers store the product information.

Note: This field is optional. Providers can enter the UR location along with product advertisement submission.

Column: providerId

Description: The unique ID which identifies a provider advertisement.

Note: The providerId is equivalent to advId in the IoAdvMaster table.

Column: providerURL

Description: The address of UR which stores the provider advertisement.

Column: PublicationDate

Description: The date of formal or informal publication of the reference paper.

Column: RangeBeginningDate

Description: The year (and optionally month, or month and day) when the temporal coverage period being described began. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: RangeEndingDate

Description: The last year (and optionally month, or month and day) of the temporal coverage period being described (GSFC AVHRR). This date represents the end date of the latest granule contained in the product. This is an SDSRV Subsystem column.

Note: MM/DD/YY format is product specific for sage_atmos_dyn, sage_atmos_comp, erbe_erp. MMDDYYYY format is product specific for LARC_FIRE, LARC_GTE

Valid Values: See 420-TP-015

Column: ReferencePaperReference

Description: Contains the unique ID of the Reference Paper issued by publisher, such as 'NOS NSG', or 'JPL Publication 91-29'.

Column: requestId

Description: The unique ID assigned to each DAAC.

Column: requestIdUnique

Description: The highest value of the new AdvId which is used currently.

Column: requestTime

Description: This is the time of the request.

Column: reviewId

Description: The unique ID which identifies the review for the given collection.

Column: RevisionDate

Description: Represents the date and possibly the time that this directory entry was created or the latest date and time of its modification or update. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: rightAdvId

Description: The advId which represents a service advertisement.

Column: ScienceReviewDate

Description: Date of last QA peer review.

Column: SensorLongName

Description: The expanded name of the sensor. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SensorShortName

Description: A sensor is defined sensory sub-component of an instrument (e.g., InstrumentShortName = ASTER, NumberofSensors = 3, SensorShortName = SWIR, SensorShortName = TR, SensorShortName = VNIR). This is an SDSRV Subsystem column.

Note: If the instrument has only one sensor or the sensor is an instrument (e.g., AVHRR), the information should go to both Sensor table and Instrument table.

Valid Values: See 420-TP-015

Column: seqNumber

Description: Sequence number of line in which the address of service UR is stored.

Column: serverType

Description: This data represents the name of the server that contains the advertising data from a particular source.

Valid Values: “SDSRV”, “SBSRV”, “DIM”

Column: serviceClass

Description: The name of the C++ class the operation belongs to. This is an SDSRV Subsystem column (ServiceClass).

Valid Values: See 420-TP-015

Column: serviceId

Description: The unique ID which identifies the Mime service advertisement. This is an SDSRV Subsystem column (ServiceId).

Note: The Mime service is the Internet (WWW) service.

Valid Values: See 420-TP-015

Column: serviceName

Description: The name of a C++ operation. This is an SDSRV Subsystem column (ServiceName).

Valid Values: See 420-TP-015

Column: serviceSignatureId

Description: The ID which identifies the signature service.

Column: serviceURId

Description: The ID which identifies the address of service UR.

Column: serviceURL

Description: The address of UR for the MIME service.

Note: MIME service is the Internet (WWW) service.

Column: ShortName

Description: This name will identify the short name associated with the collection or granule. This includes the ECS Technical Baseline product names (e.g., CERO2, MOD12). This is the official reference name used in identifying the contents of the data collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: siteId

Description: The numeric ID assigned to the DAAC.

Note: The value of the siteId for the home DAAC is always 1. The value of each remote DAAC's siteId varies from DAAC to DAAC

Column: siteName

Description: The name of the DAAC.

Column: SouthBoundingCoordinate

Description: Southern-most limit of coverage expressed in geodetic latitude.

Valid Values: [-90.0 <= South Bounding Coordinate <=90.0], South Sounding Coordinate <= North Bounding Coordinate]

Column: SpatialCoverageType

Description: This attribute denotes whether the locality/coverage requires horizontal, vertical, or both in the spatial domain and coordinate system definitions. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SpatialKeyword

Description: This attribute specifies a word or phrase that summarizes the spatial regions covered by the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: startDate

Description: Date that the advertisement become valid. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: StateProvince

Description: The state or province of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: StreetAddress

Description: The street of the address of the point of contact. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: SuggestedUsage

Description: This attribute describes how this collection or granule may be best used to support earth science/global change research. This is an SDSRV Subsystem column.

Number of organization or individual who is point of contact. The general format of the number includes country, area, and STD codes, as required for the full telephone number. Multi-extensions should be single entries rather than part of a single entry text.

Valid Values: See 420-TP-015

Column: TelephoneNumber

Description: The telephone number of the organization or individual identified as the point of contact. The general format of the number includes country, area and STD codes, as required for the full telephone number. Multi-extensions should be single entries rather than part of a single entry text. This is an SDSRV Subsystem column

Valid Values: See 420-TP-015

Column: TelephoneNumberType

Description: The type of telephone number being provided in this instance of the phone number, in order to reach the organization or individual who serves as a point of contact. Voice number is used to speak to the organization or individual, the TDD/TYY number which hearing-impaired can converse with organization or individual, or the fax/csimile of the organization or individual. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: TemporalKeyword

Description: This attribute specifies a word or phrase which serves to summarize the temporal characteristics referenced in the collection. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: Title

Description: The full title of the reference paper. This is an SDSRV Subsystem column.

Valid Values: See 420-TP-015

Column: title

Description: The full title of the advertisement.

Column: type

Description: The type of request.

Column: uniqueId

Description: The IDs which uniquely identify an advertisement within the local site.

Column: upperTitle

Description: The name of the advertisement in upper case characters.

Column: VersionDescription

Description: Version description of the data collection.

Column: VersionID

Description: Version identifier of the data collection. This is an SDSRV Subsystem column.

Note: The default is "1"

Valid Values: See 420-TP-015

Column: WestBoundingCoordinate

Description: Western-most coordinate of the limit of coverage expressed in geodetic longitude.

Valid Values: [-180.0 <= West Bounding Coordinate <= 180.0]

Column: who

Description: The name of the person or organization who made the request.

Column: why

Description: The explanation of why the change is being requested.

3.1.4 Column Domains

Domains specify the ranges of values allowed for a given column within the database table. Sybase supports the definition of specific domains to further limit the format of data for a given column. Sybase domains are, in effect, user-defined data types

Domain: AdvIdType**Domain: SiteIdType****Domain: UniqueIdType**

3.1.5 Column Default Values

A default value is used to supply a value for a column when one is not defined at row insert time. Values default to null values where no other specification is identified for the column in Section 3.1.3; reference 311-CD-107-002 for those columns marked as replicated columns from the SDSRV Subsystem.

3.1.6 Rules

Sybase supports the definition of rules. Rules provide a means for enforcing domain constraints on a given column or combination of columns. Currently, there are no rules defined in the IOS databases.

3.1.7 Views

Sybase allows the definition of views as a means of limiting an application or users access to data in the database. Views create a logical database table from columns found in one or more database tables. There are currently no views defined within the IOS database.

3.1.8 Declarative Integrity Constraints

Sybase allows the enforcement of referential integrity via the use of declarative integrity constraints. Integrity constraints allow the SQL server to enforce primary and foreign key integrity checks automatically without requiring programming. Sybase constraints support “restrict-only” operations. This means that a row can not be deleted or the key values modified if there are rows in other database tables having a foreign key dependency on that row. Cascade delete and update operations cannot be performed if a declarative integrity constraint has been used. The following information describes the declarative integrity constraints contained in the IOS database.

Table 3-56. Dependencies on Table: *IoAdAdvMaster*

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAdvXref	advId	RightAdvId
IoAdApprovedAdv	advId	AdvId
IoAdContactAdvXref	advId	AdvId
IoAdProductAdv	advId	ProductId
IoAdProviderAdv	advId	ProviderId
IoAdRequest	advId	NewAdvId
IoAdAdvXref	advId	LeftAdvId
IoAdServiceAdv	advId	ServiceId

Table 3-57. Dependencies on Table: IoAdAnalysisSource

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAnalysisCollectionXref	AnalysisShortName	AnalysisShortName

Table 3-58. Dependencies on Table: IoAdCampaign

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdCampaignCollectionXref	CampaignShortName	CampaignShortName

Table 3-59. Dependencies on Table: IoAdContact

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdContactAddress	contactId	contactId
IoAdContactEmail	contactId	contactId
IoAdContactOrganization	contactId	organizationId
IoAdContactPerson	contactId	contactId
IoAdContactTelephone	contactId	contactId
IoAdAlgorithmPackage	contactId	contactId
IoAdContactAdvXref	contactId	contactId

Table 3-60. Dependencies on Table: IoAdContactOrganization

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdPersonOrganizationXref	organizationId	organizationId

Table 3-61. Dependencies on Table: *IoAdContactPerson*

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdPersonOrganizationXref	contactId	contactId

Table 3-62. Dependencies on Table: *IoAdECSCollection*

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAnalysisCollectionXref	collectionId	collectionId
IoAdBoundingRectangle	collectionId	collectionId
IoAdCampaignCollectionXref	collectionId	collectionId
IoAdECSCollectionKeyword	collectionId	collectionId
IoAdMultipleTypeCollection	collectionId	collectionId
IoAdRangeDateTime	collectionId	collectionId
IoAdRefPaperCollectionXref	collectionId	collectionId
IoAdSingleTypeCollection	collectionId	collectionId
IoAdSpatial	collectionId	collectionId
IoAdSpatialKeywordClass	collectionId	collectionId
IoAdTemporalKeywordClass	collectionId	collectionId
IoAdInstrumentCollectionXref	collectionId	collectionId
IoAdPlatformCollectionXref	collectionId	collectionId
IoAdSensorCollectionXref	collectionId	collectionId

Table 3-63. Dependencies on Table: *IoAdGroup*

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAdvMaster	groupId	advGroupId

Table 3-64. Dependencies on Table: IoAdInstrument

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdInstrumentCollectionXref	InstrumentShortName	InstrumentShortName

Table 3-65. Dependencies on Table: IoAdMultipleTypeCollection

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdCollectionXref	collectionId	collectionIdRef
IoAdCollectionXref	collectionId	collectionIdAgg

Table 3-66. Dependencies on Table: IoAdPlatform

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdPlatformCollectionXref	PlatformShortName	PlatformShortName

Table 3-67. Dependencies on Table: IoAdProductAdv

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdECSCollection	productId	collectionId

Table 3-68. Dependencies on Table: IoAdProviderAdv

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdServiceAdv	providerId	providerId
IoAdProductAdv	providerId	providerId

Table 3-69. Dependencies on Table: IoAdReferencePaper

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdRefPaperCollectionXref	ReferencePaperReference	ReferencePaperReference

Table 3-70. Dependencies on Table: IoAdSensor

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdSensorCollectionXref	SensorShortName	SensorShortName

Table 3-71. Dependencies on Table: IoAdServiceAdv

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdInstallableServiceAdv	serviceId	serviceId
IoAdMimeServiceAdv	serviceId	serviceId
IoAdSignatureServiceAdv	serviceId	serviceId

Table 3-72. Dependencies on Table: IoAdSignatureServiceAdv

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAdvUR	serviceId	serviceId
IoAdSignature	serviceId	serviceId

Table 3-73. Dependencies on Table: IoAdSingleTypeCollection

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAlgorithmPackage	collectionId	collectionId
IoAdReview	collectionId	collectionId
IoAdCollectionXref	collectionId	collectionIdRef

Table 3-74. Dependencies on Table: IoAdSiteInfo

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdContact	siteld	siteld
IoAdGroup	siteld	Siteld
IoAdAdvMaster	siteld	Siteld
IoAdRequest	siteld	Siteld

Table 3-75. Dependencies on Table: IoAdSpatial

Reference by List

Referenced by	Primary Key	Foreign Key
IoAdAltitudeResolution	collectionId	collectionId
IoAdDepthResolution	collectionId	collectionId

3.1.9 Triggers

The Sybase RDBMS supports the enforcement of business rules for data integrity via the use of trigger code. A trigger is defined as a set of activities or checks that are performed automatically by the RDBMS whenever an update command occurs for a given database table. The term *update* is defined to identify those database commands that result in inserting (adding) a row, changing a column within an existing row, and deleting a row. When creating a trigger, one specifies for which type of *update* the trigger will be automatically performed insert, update, delete, or a combination of the three. Insert triggers are performed automatically whenever a row is inserted into the table. Update triggers are performed automatically whenever a row is updated in the table. Delete triggers are performed automatically whenever a row is deleted from the table.

Triggers are implemented at the database table level. Table 3-76 below provides an alphabetically ordered set of triggers used to assure data integrity for the IOS database. This table includes the trigger name, the database table it is related to, and a brief description of the trigger. Table 3-76 is followed immediately by trigger source code listings. Each trigger listing is found in the same order as in Table 3-76.

Table 3-76. Trigger Descriptions

Database Table	Trigger Name	Trigger Type	Description
IoAdApprovedAdv	IoAdApprovedAdvDuplicateCheck	Insert Update	Create an error return if the approved ad description conflicts with logically unique information in an existing ad.
IoAdContactAddress	IoAdGenerateAddressId	Insert	Trigger that generates Address Id which is maximum Address Id + 1.
IoAdContact	IoAdGenerateContactId	Insert	Trigger that generates Contact Id which is concatenation of Site Id + Contact Id
IoAdGroup	IoAdGenerateGroupId	Insert	Trigger that generates Group Id which is Site Id + Group Id.
IoAdECSCollectionKeyword	IoAdGenerateKeywordId	Insert	Trigger that generates Keyword Id which is maximum Keyword Id + 1.
IoAdReview	IoAdGenerateReviewId		Trigger that generates Review Id which is maximum Review Id + 1.
IoAdRequest	IoAdRequestDuplicateCheck	Insert	Trigger that generates Request Id which is Site Id + Request Id.-Raise an error if this request's ad description conflicts with logically unique information in an existing approved ad.
IoAdAdvMaster	IoAdSetMaster	Insert	Sets the Advertisement Id which is Site Id + Unique Id and sets the UpperTitle column to uppercase of Title column in the database table
IoAdAdvMaster	IoAdUpperTitle	Update	Sets the UpperTitle column to uppercase.

3.1.9.1 Trigger: IoAdApprovedAdvDuplicateCheck

Trigger Code

```
--||||||||||||||||||||||||||||||  
-- Name : IoAdApprovedAdvDuplicateCheck  
--  
-- Description:  
--  
-- Raise an error if this approved ad description  
-- conflicts with logically  
-- unique information in an existing ad.  
--  
-- Implementation Notes:  
--||||||||||||||||||||||
```

create trigger IoAdApprovedAdvDuplicateCheck on IoAdApprovedAdv
for insert, update as

```
declare @id int  
declare @approved bit  
  
select @id = advId  
from inserted  
  
exec IoAdAdvDuplicateCheck @old=@id, @new=@id, @approved=1  
return  
go
```

3.9.1.2 Trigger: IoAdGenerateAddressId

Trigger Code

```
--||||||||||||||||||||||||||  
-- Name: IoAdGenerateAddressId  
--  
-- Description:  
--  
-- Trigger that generates Address Id which is maximum Address Id + 1  
--  
-- Implementation Notes:  
--||||||||||||||||||||||
```

create trigger IoAdGenerateAddressId on IoAdContactAddress
for insert as
begin

```
declare @maxId      AdvIdType  
  
select @maxId = max(addressId)  
      from IoAdContactAddress  
      if @maxId = NULL
```

```

        select  @maxId = 0

        update IoAdContactAddress
            set addressId = @maxId + 1
            where addressId = 0
    end
return
go

```

3.1.9.3 Trigger: IoAdGenerateContactId

Trigger Code

```
--|||||||||||||||||||||||||||||||||
-- Name: IoAdGenerateContactId
--
-- Description:
--
-- Trigger that generates Contact Id which is Site Id + Contact Id
--
-- Implementation Notes:
--|||||||||||||||||||||||||||||||
```

```

create trigger IoAdGenerateContactId on IoAdContact
for insert as
begin
    declare @siteld      SitIdType,
            @uniqueId   AdvIdType,
            @contactId  AdvIdType
    declare @error       int
    begin
        select @contactId = 0, @siteld = 0, @uniqueId = 0
        exec IoAdGetContactKey @siteld = @siteld output,
             @uniqueId = @uniqueId output,
             @contactId = @contactId output,
             @error = @error output
        if @error != 0
            return
        update IoAdContact
            set uniqueId = @uniqueId,
                contactId = @contactId,
                sitId = @siteld
                where uniqueId = 0
    end
end
return
go

```

3.1.9.4 Trigger: IoAdGenerateGroupId

Trigger Code

```
--||||||||||||||||||||||||||||||  
-- Name: IoAdGenerateGroupId  
--  
-- Description:  
--  
-- Trigger that generates Group Id which is Site Id + Group Id  
--  
-- Implementation Notes:  
--||||||||||||||||||||||||||
```

```
create trigger IoAdGenerateGroupId on IoAdGroup  
for insert as  
begin  
    declare @siteld      SitIdType,  
            @uniqueId     AdvIdType,  
            @groupId       AdvIdType  
    declare @error        int  
    begin  
        select @groupId = 0, @siteld = 0, @uniqueId = 0  
        exec IoAdGetGroupKey @siteld = @siteld output,  
                           @uniqueId = @uniqueId output,  
                           @groupId = @groupId output,  
                           @error = @error output  
        if @error != 0  
            return  
        update IoAdGroup  
            set uniqueId = @uniqueId,  
                groupId = @groupId,  
                siteld = @siteld  
            where uniqueId = 0  
    end  
end  
return  
go
```

3.1.9.5 Trigger: IoAdGenerateKeywordId

Trigger Code

```
--||||||||||||||||||||||||||  
-- Name: IoAdGenerateKeywordId  
--  
-- Description:  
--  
-- Trigger that generates Keyword Id which is maximum Keyword Id + 1  
--  
-- Implementation Notes:  
--||||||||||||||||||||||
```

```

create trigger IoAdGenerateKeywordId on IoAdECSCollectionKeyword
for insert as
begin

declare @maxId      AdvIdType

select @maxId = max(keywordId)
       from IoAdECSCollectionKeyword
if @maxId = NULL
    select @maxId = 0

update IoAdECSCollectionKeyword
    set keywordId = @maxId + 1
    where keywordId = 0
end
return
go

```

3.1.9.6 Trigger: IoAdGenerateReviewId

Trigger Code

```

--|||||||||||||||||||||||||||||||||
-- Name: IoAdGenerateReviewId
--
-- Description:
--
-- Trigger that generates Review Id which is maximum Review Id + 1
--
-- Implementation Notes:
--|||||||||||||||||||||||||||||

```

create trigger IoAdGenerateReviewId on IoAdReview
for insert as

```

begin

declare @maxId      AdvIdType

select @maxId = max(reviewId)
       from IoAdReview
if @maxId = NULL
    select @maxId = 0

update IoAdReview
    set reviewId = @maxId + 1
    where reviewId = 0
end
return
go

```

3.1.9.7 Trigger: loAdRequestDuplicateCheck

Trigger Code

```
--||||||||||||||||||||||||||||||  
--  
-- Name : loAdRequestDuplicateCheckANDGenerateRequestId  
--  
-- Description:  
--  
-- Trigger that generates Request Id which is Site Id + Request Id  
-- Raise an error if this request's ad  
-- description conflicts with logically  
-- unique information in an existing approved ad.  
--  
-- Implementation Notes:  
--||||||||||||||||||||||||||
```

create trigger loAdRequestDuplicateCheck on loAdRequest
for insert, update as

```
begin  
    declare @siteld      SitIdType,  
            @uniqueId     AdvIdType,  
            @requestId    AdvIdType  
    declare @error        int  
-- only do that for insert, delete later ...  
    if ( select count(*) from deleted ) = 0  
    begin  
        select @requestId = 0, @siteld = 0, @uniqueId = 0  
        exec loAdGetRequestKey @siteld = @siteld output,  
                            @uniqueId = @uniqueId output,  
                            @requestId = @requestId output,  
                            @error = @error output  
        if @error != 0  
            return  
        update loAdRequest  
            set uniqueId = @uniqueId,  
                requestId = @requestId,  
                sitId = @siteld  
            where uniqueId = 0  
    end  
end  
--||||||||||||||||||||||||||
```

```
declare @kind int, @old int, @new int  
  
-- Processing is based on the kind of request  
--  
select @kind = type  
from inserted  
  
-- We need to call AdvDuplicateCheck with two IDs, the ID of  
-- the ad being inserted (@new) and possibly the ID of the ad it
```

```

-- is replacing (@old). If it is not replacing anything, old should
-- equal new.

if (@kind = 1)                                -- Addition
begin
    select @new = newAdvId
    from inserted

declare @approved bit
exec IoAdAdvDuplicateCheck @old=@new, @new=@new, @approved=0
end
else if (@kind = 2)                            -- Deletion
begin
    return
end
else if (@kind = 3)                            -- Modification
begin
    select @new = newAdvId
    from inserted

    select @old = oldAdvId
    from inserted

    exec IoAdAdvDuplicateCheck @old=@old, @new=@new, @approved=0
end
else
begin
    raiserror 30010 "Unknown Request Type in IoAdRequestDuplicateCheck"
    return
end

return
go

```

3.1.9.8 Trigger: IoAdSetMaster

Trigger Code

```

--|||||||||||||||||||||||||||||||||
-- Name: IoAdSetMaster
--
-- Description:
--
-- Trigger for IoAdSetMaster insert. It sets
-- Advertisement Id which is Site Id + Unique Id
-- It also sets the UpperTitle column to the uppercase
-- of Title column in the IoAdAdvMaster table
--
-- Implementation Notes:
--|||||||||||||||||||||||||||||

```

create trigger IoAdSetMaster on IoAdAdvMaster

```

for insert as
begin
    declare @siteld      SitIdType,
            @uniqueId   AdvIdType,
            @advId       AdvIdType
    declare @error        int

        select @advId = 0, @sitId = 0, @uniqueId = 0
        exec IoAdGetAdvMasterKey @sitId = @sitId output,
                         @uniqueId = @uniqueId output,
                         @advId = @advId output,
                         @error = @error output
        if @error != 0
            return
        update IoAdAdvMaster
            set uniqueId = @uniqueId,
                advId = @advId,
                sitId = @sitId
            where uniqueId = 0

        update IoAdAdvMaster
            set upperTitle = upper(title)
            where advId = @advId
end
return
go

```

3.1.9.9 Trigger: IoAdUpperTitle

Trigger Code

```

--/////////////////////////////////////////////////////////////////////////
-- Name: IoAdUpperTitle
--
-- Description:
--
-- Trigger that sets the UpperTitle column to the uppercase
-- of Title column in the IoAdAdvMaster table
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////////////////////
create trigger IoAdUpperTitle on IoAdAdvMaster
for update as
begin
    update IoAdAdvMaster
        set m.upperTitle = upper(m.title)
        from IoAdAdvMaster m, inserted i
        where m.advId = i.advId
end
return
go

```

3.1.10 Stored Procedures

In addition to triggers, the Sybase RDBMS supports the implementation of stored procedures for the purpose of enforcing business rules that ensure data integrity within the database. Stored procedures are parsed and compiled SQL code that reside in the database and may be called by name by an application, trigger or another stored procedure. Table 3-77 contains a summary listing of the stored procedures in the IOS database including the procedure name and a brief definition of the purpose of the procedure. Each stored procedure within the IOS database is listed following Table 3-77.

Table 3-77. Summary List of Stored Procedures (1 of 2)

Procedure Name	Description	Calls	Called By
datawarning	Notify the DBA when the data segment threshold is in danger of being crossed.	Not applicable	Sybase when data segment threshold exceeded.
IoAdAdvDuplicateCheck	Check if the new ad (@new)'s data duplicates any existing approved ad- except for an older version of it (@old),	IoAdProviderDuplicateCheck IoAdProductDupChkForApproved IoAdDupChkForRequest IoAdSigServiceDuplicateCheck IoAdServiceDuplicateCheck	IOS custom code
IoAdCreateCompositeId	Generate Composite Key from siteld and uniqueld.	Not applicable	IOS custom code
IoAdGetAdvMasterKey	Generate key AdvId from siteld and uniqueld.—Raise an error if update and select from IoAdAdvMasterKey-table not successful	Not applicable	IOS custom code
IoAdGetContactKey	Generate key ContactId from siteld and uniqueld.-- Raise an error if update and select from IoAdContactKey-table not successful	Not applicable	IOS custom code

Table 3-77. Summary List of Stored Procedures (2 of 2)

Procedure Name	Description	Calls	Called By
IoAdGetGroupKey	Generate key GroupId from siteld and uniqueld. Raise an error if update and select from IoAdGroupKey- table not successful	Not applicable	IOS custom code
IoAdGetRequestKey	Generate key RequestId from siteld and uniqueld. Raise an error if update and select from IoAdRequestKey- table not successful	Not applicable	IOS custom code
IoAdProductDupChkForApproved	Procedure that detects the duplication of key fields in this entity.	Not applicable	IoAdAdvDuplicateCheck
IoAdProductDupChkForRequest	Procedure that detects the duplication of key fields in this entity.	Not applicable	IoAdAdvDuplicateCheck
IoAdProviderDuplicateCheck	Procedure that detects the duplication of key fields in this entity.	Not applicable	IoAdAdvDuplicateCheck
IoAdServiceDuplicateCheck	Procedure that detects the duplication of key fields in this entity.	Not applicable	IoAdAdvDuplicateCheck
IoAdSigServiceDuplicateCheck	Procedure that detects the duplication of key fields in this entity.	Not applicable	IoAdAdvDuplicateCheck
logdump	Dumps the log when the log segment threshold is crossed.	Not applicable	Sybase when log segment threshold exceeded.
logwarning	Notifies the DBA when the log segment threshold is in danger of being crossed.	Not applicable	Sybase when database reaches specified percentage of log segment size.
ProcDeleteCollection	Deletes a collection based on ESDT shortname and version	Not applicable	IOS custom code.

3.1.10.1 Procedure: datawarning

Code

```
CREATE PROCEDURE datawarning
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS
DECLARE @date datetime
    SELECT @date = getdate()
PRINT "DATA/INDEX SEGMENT WARNING: database '%1!', threshold '%2!', segment '%3!',
date/time '%4!'",
    @dbname, @space_left, @segmentname, @date
go
```

3.1.10.2 Procedure: IoAdAdvDuplicateCheck

Code

```
--/////////////////////////////////////////////////////////////////////////
-- Name : IoAdAdvDuplicateCheck
--
-- Description:
--
-- Check if the new ad (@new)'s data duplicates any existing approved ad
-- except for an older version of it (@old), which it of course duplicates.
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////////////////////
```

```
create proc IoAdAdvDuplicateCheck
(@old int = null,
 @new int = null,
 @approvedCheck bit = 0)
as
    -- Validate arguments
    --
    if (@old is null OR @new is null )
    begin
        raiserror 30010 "IDs not supplied to procedure IoAdAdvDuplicateCheck"
        return
    end
```

```

-- Determine ad's type
--
declare @typeStr varchar(80)
select @typeStr = advType
from
    loAdAdvMaster tm
where
    tm.advId = @new

-- Call right routine based on ad's type
--
if (@typeStr = 'ad.provider')
    exec loAdProviderDuplicateCheck @old=@old, @new=@new
else if (@typeStr = 'ad.product' and @approvedCheck = 1)
    exec loAdProductDupChkForApproved @old=@old, @new=@new
else if (@typeStr = 'ad.product' and @approvedCheck = 0)
    exec loAdProductDupChkForRequest @old=@old, @new=@new
else if (@typeStr = 'ad.service.sig')
    exec loAdSigServiceDuplicateCheck @old=@old, @new=@new
else if (@typeStr like 'ad.service%')
    exec loAdServiceDuplicateCheck @old=@old, @new=@new
else
begin
    raiserror 30010 "Unknown Type of Ad in procedure loAdAdvDuplicateCheck"
    return
end
return
go

```

3.1.10.3 Procedure: loAdCreateCompositelD

Code

```
--create schema authorization dbo
```

```
--///////////////////////////////
-- Name : loAdCreateCompositelD
--
-- Description:
--
-- Generate Composite Key from sitelD and uniquelD.
--
-- Implementation Notes:
--/////////////////////////////
```

```
create proc loAdCreateCompositelD
    (@id1 SitelDType output,
```

```

@id2 UniqueIdType,
@id3 AdvIdType      output,
@error int          output)
as
declare @siteStr           varchar(2),
        @uniqueStr       varchar(6),
        @idStr           varchar(8)

select @id1 = sitId
      from IoAdSiteInfo
     where local = 1

select @error = @@error
if @error != 0
    return
select @siteStr = right
        ("00" + ltrim(convert(varchar(2), @id1)), 2)

select @idStr = right
        ("000000" + ltrim(convert(varchar(6), @id2)), 6)

select @id3 = convert(numeric(8,0), @siteStr + @idStr)

return
go

```

3.1.10.4 Procedure: IoAdGetAdvMasterKey

Code

```

--/////////////////////////////////////////////////////////////////////////
-- Name : IoAdGetAdvMasterKey
--
-- Description:
--
-- Generate key AdvId from sitId and uniqueId
-- Raise an error if update and select from IoAdAdvMasterKey
-- table not successful
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////////////////////

create proc IoAdGetAdvMasterKey
(@sitId      SitIdType output,
@uniqueId   AdvIdType output,
@advId      AdvIdType output,
@error      int      output)
as
begin transaction

-- Check the MasterAdv Key table

```

```

if (select count(*) from IoAdAdvMasterKey ) != 1
begin
    declare @maxId      UniqueldType
    delete from IoAdAdvMasterKey
    select @maxId = max(uniqueld)
        from IoAdAdvMaster g, IoAdSiteInfo s
        where s.local = 1 and s.siteld = g.siteld
    if @maxId = NULL
        select @maxId = 0
    insert into IoAdAdvMasterKey (advIdUnique) values (@maxId)
end

-- Generate key AdvId

begin
update IoAdAdvMasterKey
    set advIdUnique = advIdUnique + 1
select @error = @@error
if @error != 0
begin
    raiserror 30515
    "IoAdAdvMasterKey not supplied to procedure IoAdGetAdvMasterKey"
    rollback transaction
    return
end
select @uniqueld = advIdUnique
    from IoAdAdvMasterKey
select @error = @@error
if @error != 0
begin
    raiserror 30515
    "IoAdAdvMasterKey not supplied to procedure IoAdGetAdvMasterKey"
    rollback transaction
    return
end
select @advId = 0, @siteld = 0
exec IoAdCreateCompositelD @id1 = @siteld output,
    @id2 = @uniqueld,
    @id3 = @advId output,
    @error = @error output
if @error != 0
begin
    raiserror 30515
    "IoAdSiteInfo not supplied to procedure IoAdGetAdvMasterKey"
    rollback transaction
    return
end
end
commit transaction
return
go

```

3.1.10.5 Procedure: IoAdGetContactKey

Code

```
--||||||||||||||||||||||||||||||  
-- Name : IoAdGetContactKey  
--  
-- Description:  
--  
-- Generate key ContactId from sitId and uniqueId  
-- Raise an error if update and select from IoAdContactKey  
-- table not successful  
--  
-- Implementation Notes:  
--|||||||||||||||||||||||||||  
  
create proc IoAdGetContactKey  
(@sitId      SitIdType output,  
 @uniqueId   AdvIdType output,  
 @contactId  AdvIdType output,  
 @error       int          output)  
as  
    begin transaction  
  
    -- Check the Contact Key table  
  
        if (select count(*) from IoAdContactKey ) != 1  
        begin  
            declare @maxId      UniqueIdType  
            delete from IoAdContactKey  
  
            select @maxId = max(uniqueId)  
                  from IoAdContact g, IoAdSiteInfo s  
                 where s.local = 1  
                   and s.sitId = g.sitId  
  
            if @maxId = NULL  
                select @maxId = 0  
            insert into IoAdContactKey (contactIdUnique) values (@maxId)  
        end  
  
    -- Generate key ContactId  
  
        begin  
            update IoAdContactKey  
                set contactIdUnique = contactIdUnique + 1  
            select @error = @@error  
            if @error != 0  
                begin  
                    raiserror 30515  
                    "IoAdContactKey not supplied to procedure IoAdGetContactKey"  
                    rollback transaction  
                    return  
                end  
            select @uniqueId = contactIdUnique
```

```

from IoAdContactKey
select @error = @@error
if @error != 0
begin
    raiserror 30515
    "IoAdContactKey not supplied to procedure IoAdGetContactKey"
    rollback transaction
    return
end
select @contactId = 0, @siteId = 0
exec IoAdCreateCompositId @id1 = @siteId output,
    @id2 = @uniqueId,
    @id3 = @contactId output,
    @error = @error output
if @error != 0
begin
    raiserror 30515
    "IoAdSiteInfo not supplied to procedure IoAdGetContactKey"
    rollback transaction
    return
end
commit transaction
return
go

```

3.1.10.6 Procedure: IoAdGetGroupKey

Code

```

--/////////////////////////////////////////////////////////////////////////
-- Name : IoAdGetGroupKey
--
-- Description:
--
-- Generate key GroupId from siteId and uniqueId
-- Raise an error if update and select from IoAdGroupKey
-- table not successful
--
-- Implementation Notes:
--/////////////////////////////////////////////////////////////////////////

```

```

create proc IoAdGetGroupKey
    (@siteId      SitIdType output,
     @uniqueId    AdvIdType output,
     @groupId     AdvIdType output,
     @error       int      output)
as
begin transaction
-- Check the Group Key table

```

```

if (select count(*) from loAdGroupKey ) != 1
begin
    declare @maxId      UniqueIdType
    delete from loAdGroupKey
    select @maxId = max(uniqueld)
        from loAdGroup g, loAdSiteInfo s
        where s.local = 1 and s.siteld = g.siteld
    if @maxId = NULL
        select @maxId = 0
    insert into loAdGroupKey (groupIdUnique) values (@maxId)
end

-- Generate key GroupId

begin
update loAdGroupKey
    set groupIdUnique = groupIdUnique + 1
select @error = @@error
if @error != 0
begin
    raiserror 30515
    "loAdGroupKey not supplied to procedure loAdGetGroupKey"
    rollback transaction
    return
end
select @uniqueld = groupIdUnique
    from loAdGroupKey
select @error = @@error
if @error != 0
begin
    raiserror 30515
    "loAdGroupKey not supplied to procedure loAdGetGroupKey"
    rollback transaction
    return
end
select @groupId = 0, @siteld = 0
exec loAdCreateCompositelid @id1 = @siteld output,
    @id2 = @uniqueld,
    @id3 = @groupId output,
    @error = @error output
if @error != 0
begin
    raiserror 30515
    "loAdSiteInfo not supplied to procedure loAdGetGroupKey"
    rollback transaction
    return
end
end
commit transaction
return
go

```

3.1.10.7 Procedure: IoAdGetRequestKey

Code

```
--||||||||||||||||||||||||||||||  
-- Name : IoAdGetRequestKey  
--  
-- Description:  
--  
-- Generate key RequestId from sitId and uniqueId  
-- Raise an error if update and select from IoAdRequestKey  
-- table not successful  
--  
-- Implementation Notes:  
--|||||||||||||||||||||||||||
```

```
create proc IoAdGetRequestKey  
(@sitId      SitIdType output,  
 @uniqueId    AdvIdType output,  
 @requestId   AdvIdType output,  
 @error       int        output)  
as  
    begin transaction  
  
    -- Check the Request Key table  
  
    if (select count(*) from IoAdRequestKey ) != 1  
    begin  
        declare @maxId      UniqueIdType  
        delete from IoAdRequestKey  
        select @maxId = max(uniqueId)  
            from IoAdRequest g, IoAdSiteInfo s  
            where s.local = 1 and s.sitId = g.sitId  
        if @maxId = NULL  
            select @maxId = 0  
        insert into IoAdRequestKey (requestIdUnique) values (@maxId)  
    end  
  
    -- Generate key RequestId  
  
    begin  
        update IoAdRequestKey  
            set requestIdUnique = requestIdUnique + 1  
        select @error = @@error  
        if @error != 0  
            begin  
                raiserror 30515  
                "IoAdRequestKey not supplied to procedure IoAdGetRequestKey"  
                rollback transaction  
                return  
            end  
        select @uniqueId = requestIdUnique  
            from IoAdRequestKey  
        select @error = @@error  
        if @error != 0
```

```

begin
raiserror 30515
"IoAdRequestKey not supplied to procedure IoAdGetRequestKey"
    rollback transaction
    return
end
select @requestId = 0, @siteId = 0
exec IoAdCreateCompositeld @id1 = @siteId output,
    @id2 = @uniqueId,
    @id3 = @requestId output,
    @error = @error output
if @error != 0
begin
raiserror 30515
"IoAdSiteInfo not supplied to procedure IoAdGetRequestKey"
    rollback transaction
    return
end
end
commit transaction
return
go

```

3.1.10.8 Procedure: IoAdProductDupChkForApproved

Code

```

--|||||||||||||||||||||||||||||||||
-- Name: IoAdProductDupChkForApproved
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--     VersionID in IoAdECSCollection
--     ShortName in IoAdECSCollection
--     o Group Id
--     o Provider Id
--     o Title
--
-- Raise an error if this there is a duplication.
--
-- Implementation Notes:
--|||||||||||||||||||||||||||||

```

```

create proc IoAdProductDupChkForApproved
(@old int = null,
 @new int = null)
as

```

```

-- Check VersionID, ShortName in IoAdECSCollection
--

declare @VersionID    int,
        @ShortName   char(8)

select @VersionID = VersionID, @ShortName = ShortName
from IoAdECSCollection
where collectionId = @new

if @@rowcount = 0
begin
    raiserror 31117 "There is no such record in Collection"
return
end

declare @group_id int, @title varchar(100), @provider int

-- First, get the group, title that we inserted into the master
--

select @group_id = advGroupId, @title = title
from IoAdAdvMaster tm
where
    tm.advId = @new

-- Then, get the title that we inserted into the master
--

select @title = title
from IoAdAdvMaster tm
where
    tm.advId = @new

-- Then, get the provider that we inserted into the product table
--

select @provider = providerId
from IoAdProductAdv tp
where
    tp.productId = @new

-- Now, compare primary keys
--

if (select count(*)
from IoAdApprovedAdv ta, IoAdAdvMaster tm, IoAdProductAdv tp,
     IoAdECSCollection tc
    where
        tm.advId = tp.productId AND

```

```

tm.advId = tc.collectionId AND
tm.advId = ta.advId AND
tm.advId != @old AND
tm.advId != @new AND
tp.providerId = @provider AND
tc.VersionID = @VersionID AND
tc.ShortName = @ShortName AND
tm.advGroupId = @group_id AND
tm.title = @title) > 0
begin
    raiserror 31117 "Attempting to insert data with same
                      group, provider, version, ShortName and title as existing entry"
    return
end
return
go

```

3.1.10.9 Procedure: IoAdProductDupChkForRequest

Code

```

--|||||||||||||||||||||||||||||||||
-- Name: IoAdProductDupChkForRequest
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--     o Group Id
--     o Provider Id
--     o Title
--
-- Raise an error if there is a duplication.
--
-- Implementation Notes:
--|||||||||||||||||||||||||||||

```

```

create proc IoAdProductDupChkForRequest
(@old int = null,
 @new int = null)
as
declare @group_id int, @title varchar(100), @provider int
-- First, get the group, title that we inserted into the master
--
select @group_id = advGroupId, @title = title
from IoAdAdvMaster tm

```

```

where
tm.advId = @new

-- Then, get the title that we inserted into the master
--

select @title = title
      from loAdAdvMaster tm
      where
            tm.advId = @new

-- Then, get the provider that we inserted into the product table
--

select @provider = providerId
      from loAdProductAdv tp
      where
            tp.productId = @new

-- Now, compare primary keys
--

if (select count(*)
from loAdApprovedAdv ta, loAdAdvMaster tm, loAdProductAdv tp
      where
            tm.advId = tp.productId AND
            tm.advId = ta.advId AND
            tm.advId != @old AND
            tm.advId != @new AND
            tp.providerId = @provider AND
            tm.advGroupId = @group_id AND
            tm.title = @title) > 0
begin
      raiserror 31117 "Attempting to insert data with same
                      group, provider, version, ShortName and title as existing entry"
      return
end
return
go

```

3.1.10.10 Procedure: IoAdProviderDuplicateCheck

Code

```
--||||||||||||||||||||||||||||||  
-- Name: IoAdProviderDuplicateCheck  
--  
-- Description:  
--  
-- Procedure that detects the duplication of key fields in  
-- this entity.  
--  
-- This entity is uniquely determined by the following composite key  
--      o Group Id  
--      o Title  
--  
-- Raise an error if this there is a duplication.  
--  
-- Implementation Notes:  
--||||||||||||||||||||||||||
```

```
create proc IoAdProviderDuplicateCheck  
(@old int = null,  
 @new int = null)  
as  
    declare @group_id int, @title varchar(100)  
  
    -- First, get the group, title that we inserted into the master  
    --  
    select @group_id = advGroupId, @title = title  
    from IoAdAdvMaster tm  
    where  
        tm.advId = @new  
    -- Now, compare primary keys, making sure to ignore old and new  
    --  
  
    if (select count(*)  
    from IoAdApprovedAdv ta, IoAdAdvMaster tm, IoAdProviderAdv tp  
    where  
        ta.advId = tm.advId AND  
        tm.advId = tp.providerId AND  
        tm.advId != @old AND  
        tm.advId != @new AND  
        tm.advGroupId = @group_id AND  
        tm.title = @title) > 0  
    begin  
        raiserror 31117 "Attempting to insert data with same  
                        group, and title as existing entry"  
        return  
    end  
return  
go
```

3.1.10.11 Procedure: IoAdServiceDuplicateCheck

Code

```
--||||||||||||||||||||||||||||||  
-- Name: IoAdServiceDuplicateCheck  
--  
-- Description:  
--  
-- Procedure that detects the duplication of key fields in  
-- this entity.  
--  
-- This entity is uniquely determined by the following composite key  
--     o Group Id  
--     o Provider Id  
--     o title  
--  
-- Raise an error if this there is a duplication.  
--  
-- Implementation Notes:  
--||||||||||||||||||||||||||
```

```
create proc IoAdServiceDuplicateCheck  
(@old int = null,  
 @new int = null)  
as  
    declare @group_id int, @title varchar(100), @provider int  
  
    -- First, get the group, title that we inserted into the master  
    --  
    select @group_id = advGroupId, @title = title  
    from IoAdAdvMaster tm  
    where  
        tm.advId = @new  
  
    -- Then, get the provider that we inserted into the product table  
    --  
    select @provider = providerId  
    from IoAdServiceAdv ts  
    where  
        ts.serviceId = @new  
  
    -- Now, compare primary keys  
    --  
    if (select count(*)  
        from IoAdApprovedAdv ta, IoAdAdvMaster tm, IoAdServiceAdv ts  
        where  
            ta.advId = tm.advId AND  
            tm.advId = ts.serviceId AND
```

```

tm.advId != @old AND
tm.advId != @new AND
tm.advGroupId = @group_id AND
ts.providerId = @provider AND
tm.title = @title) > 0
begin
    raiserror 31117 "Attempting to insert data with same
                     group, provider, and title as existing entry"
    return
end

return
go

```

3.1.10.12 Procedure: loAdSigServiceDuplicateCheck

Code

```

--|||||||||||||||||||||||||||||||||
-- Name: loAdSigServiceDuplicateCheck
--
-- Description:
--
-- Procedure that detects the duplication of key fields in
-- this entity.
--
-- This entity is uniquely determined by the following composite key
--     o Group Id
--     o Provider Id
--     o Internal Name
--
-- Raise an error if there is a duplication.
--
-- Implementation Notes:
--|||||||||||||||||||||||||

```

```

create proc loAdSigServiceDuplicateCheck
(@old int = null,
 @new int = null)
as
    declare @group_id int, @internalName varchar(100), @provider int

    -- First, get the group that we inserted into the master
    --

    select @group_id = advGroupId
    from loAdAdvMaster tm
    where
        tm.advId = @new

```

```

-- Then, get the internal Name that we inserted into the master
--
select @internalName = internalName
from IoAdSignatureServiceAdv tss
where
    tss.serviceld = @new

-- Then, get the provider that we inserted into the product table
--
select @provider = providerId
from IoAdServiceAdv ts
where
    ts.serviceld = @new

-- Now, compare primary keys
--
if (select count(*)
from IoAdApprovedAdv ta, IoAdAdvMaster tm, IoAdServiceAdv ts,
     IoAdSignatureServiceAdv tss
where
    ta.advId = tm.advId AND
    tm.advId = ts.serviceld AND
    tm.advId = tss.serviceld AND
    tm.advId != @old AND
    tm.advId != @new AND
    tm.advGroupId = @group_id AND
    ts.providerId = @provider AND
    tss.internalName = @internalName) > 0
begin
    raiserror 31117 "Attempting to insert data with same
                      group, provider, and internalName as existing entry"
    return
end

-- Now we make sure that the service criteria also are met
--
exec IoAdServiceDuplicateCheck @old=@old, @new=@new

return
go

```

3.1.10.13 Procedure: logdump

Code

```
CREATE PROCEDURE logdump
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS

DECLARE @devname varchar(100),
@before_size int,
@after_size int,
@before_time datetime,
@after_time datetime,
@error int,
@date char(8),
@time char(4)

/* get the time and log size just before the dump */

SELECT @before_time = getdate(), @before_size = reserved_pgs(id, doampg)
FROM sysindexes
WHERE sysindexes.name = "syslogs"
PRINT "LOG DUMP: database '%1!', threshold '%2!'", @dbname, @space_left

/* get current date and time in format suitable for a filename suffix */

SELECT @date = datename(yy,getdate())+
right('0'+ltrim(convert(varchar(2),
datepart(mm,getdate())))),2)+
right('0'+ltrim(convert(varchar(2),
datepart(dd,getdate())))),2)

SELECT @time = right('0'+ltrim(convert(char(2),
datepart(hh,getdate())))),2)+
right('0'+ltrim(convert(char(2),
datepart(mi,getdate())))),2)

/* dump device is a file in a mode-dependent directory owned by sybase */

SELECT @devname = "/usr/ecs/FRT/COTS/sybase/sybase.dumps/"
+ @dbname + "_tran_dmp." + @date + @time

/* dump the transaction log to the specified device */
```

```

DUMP TRANSACTION @dbname TO @devname
SELECT @error = @@error
IF @error != 0
BEGIN
PRINT "LOG DUMP ERROR: %1!", @error
RETURN @error
END

/* get size of log and time after dump */

SELECT @after_time= getdate(), @after_size =
       reserved_pgs(id, doampg)
FROM sysindexes
WHERE sysindexes.name = "syslogs"

/* print message to error log */

PRINT "LOG DUMPED TO: device '%1!', @devname
PRINT "LOG DUMP PAGES: Before: '%1!', After '%2!',
      @before_size, @after_size
PRINT "LOG DUMP TIME: %1!, %2!",
      @before_time, @after_time
go

```

3.1.10.14 Procedure: logwarning

Code

```

CREATE PROCEDURE logwarning
    @dbname varchar(30),
    @segmentname varchar(30),
    @space_left int,
    @status int
AS
DECLARE @date datetime
    SELECT @date = getdate()
PRINT "LOG WARNING: database '%1!', threshold '%2!', date/time '%3!'",
    @dbname, @space_left, @date
go

```

3.1.10.15 Procedure: ProcDeleteCollection

Code

```
/*#####
/*
/* Name      : ProcDeleteCollection      */
/* Author    : Greg Dobbins            */
/* Description : Deletes a collection based on ESDT shortname and version */
/* Date      : October 9, 1998          */
/*
/* HISTORY
/* Date      Who        What           */
/* -----   -----   ----- */
/* 10/15/98  Greg Dobbins  Code inspection */
/* 11/06/98  Greg Dobbins  Unit testing completed */
/*
/*#####
CREATE PROCEDURE ProcDeleteCollection (@shortname char(8), @version int)
AS

DECLARE @collection numeric(8,0)
DECLARE @analysisShortName char(20)
DECLARE @campaignShortName char(20)
DECLARE @platformShortName char(20)
DECLARE @instrumentShortName char(20)
DECLARE @sensorShortName char(20)
DECLARE @counter int
DECLARE @contactId numeric(8,0)
DECLARE @refPaper char(20)
DECLARE @advId numeric(8,0)

/* 1. Match short name and version with collectionId in IoAdECSCollection */

SELECT @counter = (SELECT count(*) FROM IoAdECSCollection WHERE
                    ShortName = @shortname AND VersionID = @version)
IF @counter > 1
BEGIN
    PRINT "There is more than one collection matching this shortname and version"
    RETURN
END

SELECT @collection = (SELECT collectionId FROM IoAdECSCollection WHERE
                    ShortName = @shortname AND VersionID = @version)
IF @collection is null
BEGIN
```

```
PRINT "No collection matching this shortname and version was found"
RETURN
END
```

```
/* 1a. Check that collection is "approved" */
```

```
IF @collection NOT IN (SELECT advId FROM IoAdApprovedAdv)
BEGIN
    PRINT "Collection %1! is not approved ... unable to delete", @collection
    RETURN
END
```

```
/* 1b. Check that collection is not a provider */
```

```
IF @collection IN (SELECT providerId FROM IoAdProviderAdv)
BEGIN
    PRINT "Collection %1! is a provider ... unable to delete", @collection
    RETURN
END
```

```
/* All deletes should succeed or fail together */
```

```
PRINT "Attempting deletion of collection %1! ... ", @collection
```

```
BEGIN TRANSACTION
```

```
/* 2. Delete this collectionId after deleting from dependent tables */
```

```
DELETE IoAdSpatialKeywordClass WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdSpatialKeywordClass failed"
    ROLLBACK TRANSACTION
    RETURN
END
```

```
DELETE IoAdTemporalKeywordClass WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdTemporalKeywordClass failed"
    ROLLBACK TRANSACTION
    RETURN
END
```

```
DELETE IoAdAltitudeResolution WHERE collectionId = @collection
```

```

IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdAltitudeResolution failed"
    ROLLBACK TRANSACTION
    RETURN
END

DELETE IoAdDepthResolution WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdDepthResolution failed"
    ROLLBACK TRANSACTION
    RETURN
END

DELETE IoAdSpatial WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdSpatial failed"
    ROLLBACK TRANSACTION
    RETURN
END

DELETE IoAdBoundingRectangle WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdBoundingRectangle failed"
    ROLLBACK TRANSACTION
    RETURN
END

DELETE IoAdECSCollectionKeyword WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdECSCollectionKeyword failed"
    ROLLBACK TRANSACTION
    RETURN
END

/* delete from table IoAdAnalysisSource rows referenced only by this collection */

DECLARE C1 CURSOR for
SELECT AnalysisShortName FROM IoAdAnalysisCollectionXref
    WHERE collectionId = @collection

```

```

OPEN C1
FETCH C1 INTO @analysisShortName
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdAnalysisCollectionXref
                       WHERE AnalysisShortName = @analysisShortName)
    DELETE IoAdAnalysisCollectionXref WHERE collectionId = @collection
        AND AnalysisShortName = @analysisShortName
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdAnalysisCollectionXref failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    IF @counter = 1
    BEGIN
        DELETE IoAdAnalysisSource WHERE AnalysisShortName = @analysisShortName
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdAnalysisCollectionXref failed"
            ROLLBACK TRANSACTION
            RETURN
        END
        END
        FETCH C1 INTO @analysisShortName
    END
    CLOSE C1
    DEALLOCATE CURSOR C1

    DELETE IoAdRangeDateTime WHERE collectionId = @collection
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdRangeDateTime failed"
        ROLLBACK TRANSACTION
        RETURN
    END
/* delete from table IoAdCampaign rows referenced only by this collection */

DECLARE C2 CURSOR for
SELECT CampaignShortName FROM IoAdCampaignCollectionXref
    WHERE collectionId = @collection
OPEN C2
FETCH C2 INTO @campaignShortName
WHILE (@@sqlstatus = 0)
BEGIN

```

```

SELECT @counter = (SELECT count(*) FROM IoAdCampaignCollectionXref
                   WHERE CampaignShortName = @campaignShortName)
DELETE IoAdCampaignCollectionXref WHERE collectionId = @collection
      AND CampaignShortName = @campaignShortName
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdCampaignCollectionXref failed"
    ROLLBACK TRANSACTION
    RETURN
END
IF @counter = 1
BEGIN
    DELETE IoAdCampaign WHERE CampaignShortName = @campaignShortName
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdCampaign failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    END
    FETCH C2 INTO @campaignShortName
END
CLOSE C2
DEALLOCATE CURSOR C2

/* delete from table IoAdPlatform rows referenced only by this collection */

DECLARE C3 CURSOR for
SELECT PlatformShortName FROM IoAdPlatformCollectionXref
      WHERE collectionId = @collection
OPEN C3
FETCH C3 INTO @platformShortName
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdPlatformCollectionXref
                       WHERE PlatformShortName = @platformShortName)
    DELETE IoAdPlatformCollectionXref WHERE collectionId = @collection
      AND PlatformShortName = @platformShortName
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdPlatformCollectionXref failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    IF @counter = 1
    BEGIN

```

```

DELETE IoAdPlatform WHERE PlatformShortName = @platformShortName
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdPlatform failed"
    ROLLBACK TRANSACTION
    RETURN
END
END
FETCH C3 INTO @platformShortName
END
CLOSE C3
DEALLOCATE CURSOR C3

/* delete from table IoAdInstrument rows referenced only by this collection */

DECLARE C4 CURSOR for
SELECT InstrumentShortName FROM IoAdInstrumentCollectionXref
    WHERE collectionId = @collection
OPEN C4
FETCH C4 INTO @instrumentShortName
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdInstrumentCollectionXref
        WHERE InstrumentShortName = @instrumentShortName)
    DELETE IoAdInstrumentCollectionXref WHERE collectionId = @collection
        AND InstrumentShortName = @instrumentShortName
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdInstrumentCollectionXref failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    IF @counter = 1
    BEGIN
        DELETE IoAdInstrument WHERE InstrumentShortName = @instrumentShortName
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdInstrument failed"
            ROLLBACK TRANSACTION
            RETURN
        END
    END
    FETCH C4 INTO @instrumentShortName
END
CLOSE C4
DEALLOCATE CURSOR C4

```

```

/* delete from table IoAdSensor rows referenced only by this collection */

DECLARE C5 CURSOR for
SELECT SensorShortName FROM IoAdSensorCollectionXref
    WHERE collectionId = @collection
OPEN C5
FETCH C5 INTO @sensorShortName
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdSensorCollectionXref
        WHERE SensorShortName = @sensorShortName)
    DELETE IoAdSensorCollectionXref WHERE collectionId = @collection
        AND SensorShortName = @sensorShortName
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdSensorCollectionXref failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    IF @counter = 1
    BEGIN
        DELETE IoAdSensor WHERE SensorShortName = @sensorShortName
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdSensor failed"
            ROLLBACK TRANSACTION
            RETURN
        END
        END
        FETCH C5 INTO @sensorShortName
    END
    CLOSE C5
    DEALLOCATE CURSOR C5

    DELETE IoAdAlgorithmPackage WHERE collectionId = @collection
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdAlgorithmPackage failed"
        ROLLBACK TRANSACTION
        RETURN
    END

    DELETE IoAdReview WHERE collectionId = @collection
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN

```

```

RAISERROR 99999 "Attempt to delete from table IoAdReview failed"
ROLLBACK TRANSACTION
RETURN
END

DELETE IoAdSingleTypeCollection WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdSingleTypeCollection failed"
    ROLLBACK TRANSACTION
    RETURN
END

/* delete from table IoAdReferencePaper rows referenced only by this collection */

DECLARE C6 CURSOR for
SELECT ReferencePaperReference FROM IoAdRefPaperCollectionXref
    WHERE collectionId = @collection
OPEN C6
FETCH C6 INTO @refPaper
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdRefPaperCollectionXref
        WHERE ReferencePaperReference = @refPaper)
    DELETE IoAdRefPaperCollectionXref WHERE collectionId = @collection
        AND ReferencePaperReference = @refPaper
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdRefPaperCollectionXref failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    IF @counter = 1
    BEGIN
        DELETE IoAdReferencePaper WHERE ReferencePaperReference = @refPaper
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdReferencePaper failed"
            ROLLBACK TRANSACTION
            RETURN
        END
    END
    FETCH C6 INTO @refPaper
END
CLOSE C6
DEALLOCATE CURSOR C6

```

```
/* Lastly, delete from IoAdECSCollection; all dependent foreign keys should already
   have been deleted by this point. */
```

```
DELETE IoAdECSCollection WHERE collectionId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
  RAISERROR 99999 "Attempt to delete from table IoAdECSCollection failed"
  ROLLBACK TRANSACTION
  RETURN
END
```

```
/* 3. Consider IoAdAdvXref where collectionId = leftAdvId. For matching
   rightAdvId, get matching advId in table IoAdAdvMaster and then matching
   serviceId in IoAdServiceAdv. Delete matches in IoAdServiceAdv. Delete
   matching serviceId in tables IoAdSignatureServiceAdv and cascade delete
   to IoAdSignature and IoAdAdvUR. */
```

```
DELETE IoAdSignature FROM IoAdSignature t1, IoAdAdvXref t2 WHERE
t1.serviceId = t2.rightAdvId AND t2.leftAdvId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
  RAISERROR 99999 "Attempt to delete from table IoAdSignature failed"
  ROLLBACK TRANSACTION
  RETURN
END
```

```
DELETE IoAdAdvUR FROM IoAdAdvUR t1, IoAdAdvXref t2 WHERE
t1.serviceId = t2.rightAdvId AND t2.leftAdvId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
  RAISERROR 99999 "Attempt to delete from table IoAdAdvUR failed"
  ROLLBACK TRANSACTION
  RETURN
END
```

```
DELETE IoAdSignatureServiceAdv FROM IoAdSignatureServiceAdv t1,
IoAdAdvXref t2 WHERE t1.serviceId = t2.rightAdvId AND t2.leftAdvId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
  RAISERROR 99999 "Attempt to delete from table IoAdSignatureServiceAdv failed"
  ROLLBACK TRANSACTION
  RETURN
END
```

```
DELETE IoAdServiceAdv FROM IoAdServiceAdv t1, IoAdAdvXref t2 WHERE
```

```

t1.serviceId = t2.rightAdvId AND t2.leftAdvId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdServiceAdv failed"
    ROLLBACK TRANSACTION
    RETURN
END

```

/* 4. For matching advId in step 3) consider matches in IoAdContactAdvXref.
Delete here only if contact info relates just to this advId. If deleting
contact, then cascade delete to dependent tables. */

```

DECLARE C7 CURSOR for
SELECT contactId FROM IoAdContactAdvXref
    WHERE advId = @collection
OPEN C7
FETCH C7 INTO @contactId
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdContactAdvXref
        WHERE contactId = @contactId)
    IF @counter = 1
    BEGIN
        DELETE IoAdContactAdvXref WHERE contactId = @contactId
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdContactAdvXref (1) failed"
            ROLLBACK TRANSACTION
            RETURN
        END
        DELETE IoAdContactTelephone WHERE contactId = @contactId
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdContactTelephone failed"
            ROLLBACK TRANSACTION
            RETURN
        END
        DELETE IoAdContactEmail WHERE contactId = @contactId
        IF @@transtate = 2 /* previous statement aborted */
        BEGIN
            RAISERROR 99999 "Attempt to delete from table IoAdContactEmail failed"
            ROLLBACK TRANSACTION
            RETURN
        END
        DELETE IoAdContactAddress WHERE contactId = @contactId
        IF @@transtate = 2 /* previous statement aborted */
    END

```

```

BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdContactAddress failed"
    ROLLBACK TRANSACTION
    RETURN
END

/* Note that a contact could be a person or an organization.
   Since we don't know which, delete from both (only one will succeed) */

DELETE IoAdPersonOrganizationXref WHERE contactId = @contactId
    OR organizationId = @contactId
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdPersonOrganizationXref failed"
    ROLLBACK TRANSACTION
    RETURN
END
DELETE IoAdContactPerson WHERE contactId = @contactId
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdContactPerson failed"
    ROLLBACK TRANSACTION
    RETURN
END
DELETE IoAdContactOrganization WHERE organizationId = @contactId
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdContactOrganization failed"
    ROLLBACK TRANSACTION
    RETURN
END
DELETE IoAdContact WHERE contactId = @contactId
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdContact failed"
    ROLLBACK TRANSACTION
    RETURN
END
FETCH C7 INTO @contactId
END
CLOSE C7
DEALLOCATE CURSOR C7

/* Whether or not the contactId is unique to this collection, delete any reference
   to this collection in the table IoAdContactAdvXref (if not already done above) */

```

```

DELETE IoAdContactAdvXref WHERE advId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdContactAdvXref (2) failed"
    ROLLBACK TRANSACTION
    RETURN
END

/* 5a. Delete from IoAdApprovedAdv, IoAdAdvDescription, and IoAdProductAdv */

DELETE IoAdApprovedAdv WHERE advId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdApprovedAdv (1) failed"
    ROLLBACK TRANSACTION
    RETURN
END

DELETE IoAdAdvDescription WHERE advId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdAdvDescription (1) failed"
    ROLLBACK TRANSACTION
    RETURN
END

DELETE IoAdProductAdv WHERE productId = @collection
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdProductAdv (1) failed"
    ROLLBACK TRANSACTION
    RETURN
END

/* 5b. Lastly, delete from IoAdAdvMaster and IoAdAdvXref */

DECLARE C8 CURSOR for
SELECT rightAdvId FROM IoAdAdvXref
    WHERE IoAdAdvXref.leftAdvId = @collection
OPEN C8
FETCH C8 INTO @advId
WHILE (@@sqlstatus = 0)
BEGIN
    SELECT @counter = (SELECT count(*) FROM IoAdAdvXref
        WHERE rightAdvId = @advId)

```

```

DELETE IoAdAdvXref WHERE leftAdvId = @collection
    AND rightAdvId = @advId
IF @@transtate = 2 /* previous statement aborted */
BEGIN
    RAISERROR 99999 "Attempt to delete from table IoAdAdvXref failed"
    ROLLBACK TRANSACTION
    RETURN
END
IF @counter = 1
BEGIN
    DELETE IoAdApprovedAdv WHERE advId = @advId
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdApprovedAdv (2) failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    DELETE IoAdAdvDescription WHERE advId = @advId
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdAdvDescription (2) failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    DELETE IoAdProductAdv WHERE productId = @advId
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdProductAdv (2) failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    DELETE IoAdAdvMaster WHERE advId = @advId
    IF @@transtate = 2 /* previous statement aborted */
    BEGIN
        RAISERROR 99999 "Attempt to delete from table IoAdAdvMaster (1) failed"
        ROLLBACK TRANSACTION
        RETURN
    END
    END
    FETCH C8 INTO @advId
END
CLOSE C8
DEALLOCATE CURSOR C8

DELETE IoAdAdvMaster WHERE advId = @collection
IF @@transtate = 2 /* previous statement aborted */

```

```
BEGIN  
    RAISERROR 99999 "Attempt to delete from table IoAdAdvMaster (2) failed"  
    ROLLBACK TRANSACTION  
    RETURN  
END  
  
COMMIT TRANSACTION  
  
PRINT "Deletion of collection %1! was successful", @collection  
  
RETURN  
go
```

3.2 Flat File Usage

A flat file is an operating system file that is read and written serially. It is generally independent with respect to other files that exist and is generally static in nature. There are cases when the implementation of certain persistent data is better suited to a flat file than to a database (e.g., system configuration data, external interface data, log files). IOS file usage is detailed in this section via file, field, and domain definitions.

3.2.1 File Descriptions

No files used.

3.2.2 Field Specifications

Not applicable.

3.2.3 Domain Definitions

Domain definitions specify the nature and valid content of fields within a file (e.g., specific values for a limited set of data, ranges of numeric data, units of measure for applicable data). This information is generally used by software to edit incoming data for validity prior to writing or changing data within the file. Use of domain values in updating (adding and changing) records within files preserves the integrity of the data within the file. This section is not applicable to IOS.

This page intentionally left blank.

4. Performance and Tuning Factors

4.1 Indexes

An index provides a means of locating a row in a database table based on the value of one or more columns, without having to scan all data in the table. An index may be arranged in ascending or descending sort order. If used appropriately, indexes can significantly decrease the time it takes to retrieve data. Sybase allows the definition of two types of indexes, clustered and non-clustered.

In a *clustered* index, the rows in a database table are physically stored in sequence--determined by the index. Clustered indexes are particularly useful, when the data is frequently retrieved in sequential order. Only one clustered index may be defined per database table as data may be stored in only one physical sequence.

Non-clustered indexes differ from their clustered counterpart, in that data is not physically stored in sorted order—newly added rows are stored at the end of the database table.

Indexes can be defined as *unique*. In this case, only one index entry (row in the indexed table) can have a given set of values for the columns comprising the index.

Primary and foreign key constraints on a table are implemented by *primary key* and *foreign key* indexes created automatically by the SQL server.

A list of all the indexes defined for database tables in the IOS database is given in Table 4-2 along with a description of each index. A key for using interpreting this listing of indexes is given in Table 4-1.

Table 4-1. Index Key

Index Type Code	Description
P	Primary Key
F	Foreign Key
U	Unique
C	Clustered
Sort	Sort order ASCending or DESCending.

Table 4-2. Indexes (1 of 3)

Table	Index Name	Column Code	P	F	U	C	Sort
IoAdAdvDescription	pk_ioadadvdescription	advid seqNumber	Y	N	Y	Y	ASC ASC
IoAdAdvMaster	pk_ioadadvmaster	advid	Y	N	Y	Y	ASC
IoAdAdvMasterKey	pk_ioadadvmasterkey	advidUnique	Y	N	Y	Y	ASC
IoAdAdvUR	pk_ioadadvur	serviceId seqNumber	Y	N	Y	Y	ASC ASC
IoAdAdvXref	pk_ioadadvxref	leftAdvid rightAdvid	Y	N	Y	Y	ASC ASC
IoAdAlgorithmPackage	pk_ioadalgorithmpackage	AlgorithmPackageName AlgorithmPackageVersion	Y	N	Y	Y	ASC ASC
IoAdAltitudeResolution	pk_ioadaltituderesolution	collectionId	Y	Y	Y	Y	ASC
IoAdAnalysisCollectionXref	pk_ioadanalysiscollectionxref	collectionId AnalysisShortName	Y	N	Y	Y	ASC ASC
IoAdAnalysisSource	pk_ioadanalysissource	AnalysisShortName	Y	N	Y	Y	ASC
IoAdApprovedAdv	pk_ioadapprovedadv	advid	Y	Y	Y	Y	ASC
IoAdBoundingRectangle	pk_ioadboundingrectangle	collectionId	Y	Y	Y	Y	ASC
IoAdCampaign	pk_ioadcampaign	CampaignShortName	Y	N	Y	Y	ASC
IoAdCampaignCollectionXref	pk_iadcampaigncollectionxref	collectionId CampaignShortName	Y	N	Y	Y	ASC ASC
IoAdCollectionXref	pk_ioadcollectionxref	collectionIdAgg collectionIdRef	Y	N	Y	Y	ASC ASC
IoAdContact	pk_ioadcontact	contactId	Y	N	Y	Y	ASC
IoAdContactAddress	pk_ioadcontactaddress	addressId contactId	Y	N	Y	Y	ASC ASC
IoAdContactAdvXref	pk_ioadcontactadvxref	advid contactId	Y	N	Y	Y	ASC ASC
IoAdContactEmail	pk_ioadcontactemail	ElectronicMailAddress	Y	N	Y	Y	ASC
IoAdContactKey	pk_ioadcontactkey	contactIdUnique	Y	N	Y	Y	ASC
IoAdContactOrganization	pk_ioadcontactorganization	organizationId	Y	Y	Y	Y	ASC

Table 4-2. Indexes (2 of 3)

Table	Index Name	Column Code	P	F	U	C	Sort
IoAdContactPerson	pk_ioadcontactperson	contactId	Y	Y	Y	Y	ASC
IoAdContactTelephone	pk_ioadcontacttelephone	TelephoneNumber	Y	N	Y	Y	ASC
IoAdDepthResolution	pk_ioaddepthresolution	collectionId	Y	Y	Y	Y	ASC
IoAdECSCollection	pk_ioadecscollection	collectionId	Y	Y	Y	Y	ASC
IoAdCollectionKeyword	pk_ioadecscollectionkeyword	keywordId collectionId	Y	N	Y	Y	ASC ASC
IoAdGroup	Group_Name_Idx	groupName	N	N	Y	Y	ASC
loadGroup	pk_ioadgroup	groupId	Y	N	Y	N	ASC
IoAdGroupKey	pk_ioadgroupkey	groupIdUnique	Y	N	Y	Y	ASC
IoAdInstallableServiceAdv	pk_ioadinstallableserviceadv	serviceId	Y	Y	Y	Y	ASC
IoAdInstrument	pk_ioadinstrument	InstrumentShortName	Y	N	Y	Y	ASC
IoAdInstrumentXref	pk_ioadinstrumentcollectionxref	collectionId InstrumentShortName	Y	N	Y	Y	ASC ASC
IoAdMimeTypeAdv	pk_ioadmimeserviceadv	serviceId	Y	Y	Y	Y	ASC
IoAdMultipleTypeCollection	pk_ioadmultipletypecollection	collectionId	Y	Y	Y	Y	ASC
IoAdPersonOrganizationXref	pk_ioadpersonorganizationxref	contactId organizationId	Y	N	Y	Y	ASC ASC
IoAdPlatform	pk_ioadplatform	PlatformShortName	Y	N	Y	Y	ASC
IoAdPlatformCollectionXref	pk_ioadplatformcollectionxref	collectionId PlatformShortName	Y	N	Y	Y	ASC ASC
IoAdProductAdv	pk_ioadproductadv	productId	Y	Y	Y	Y	ASC
IoAdProviderAdv	pk_ioadprovideradv	providerId	Y	Y	Y	Y	ASC
IoAdRangeDateTime	pk_ioadrangedatetime	collectionId RangeBeginningDate RangeEndingDate	Y	N	Y	Y	ASC ASC ASC
IoAdReferencePaper	pk_ioadreferencepaper	ReferencePaperReference	Y	N	Y	Y	ASC
IoAdReferencePaperCollectionXref	pk_ioadrefpapercollectionxref	collectionId ReferencePaperReference	Y	N	Y	Y	ASC ASC

Table 4-2. Indexes (3 of 3)

Table	Index Name	Column Code	P	F	U	C	Sort
IoAdRequest	pk_ioadrequest	uniqueId siteId	Y	N	Y	Y	ASC ASC
IoAdRequestKey	pk_ioadrequestkey	requestIdUnique	Y	N	Y	Y	ASC
IoAdReview	pk_ioadreview	reviewId collectionId	Y	N	Y	Y	ASC ASC
IoAdSensor	pk_ioadsensor	SensorShortName	Y	N	Y	Y	ASC
IoAdSensorCollectionXref	pk_ioadsensorcollectionxref	collectionId SensorShortName	Y	N	Y	Y	ASC ASC
IoAdServiceAdv	pk_ioadserviceadv	serviceId	Y	Y	Y	Y	ASC
IoAdSignature	pk_ioadsignature	serviceId seqNumber	Y	N	Y	Y	ASC ASC
IoAdSignatureServiceAdv	pk_ioadsignatureserviceadv	serviceId	Y	Y	Y	N	ASC
IoAdSignatureServiceAdv	Signature_InternalName_Idx	internalName	N	N	N	Y	ASC
IoAdSingleTypeCollection	pk_ioadsingletypecollection	collectionId	Y	Y	Y	Y	ASC
IoAdSiteInfo	pk_ioadsiteinfo	siteId	Y	N	Y	Y	ASC
IoAdSpatial	pk_ioadspatial	collectionId	Y	Y	Y	Y	ASC
IoAdSpatialKeywordClass	pk_ioadspatialkeywordclass	collectionId SpatialKeyword	Y	N	Y	Y	ASC ASC
IoAdTemporalKeywordClass	pk_ioadtemporalkeywordclass	collectionId TemporalKeyword	Y	N	Y	Y	ASC ASC

4.2 Segments

Sybase supports the declaration of segments. A segment is a named pointer to a storage device(s). Segments are used to physically allocate a database object to a particular storage device. Segments defined for the IOS and all other subsystem databases are described in Table 4-3.

Table 4-3. Segment Descriptions

Segment Name	Description
default	Default data segment used if no other segment specified in the create statement.
logsegment	SYSLOGS, Transaction Logs
systemsegment	System tables and indexes.
IOSOPSDAT01	IOS OPS mode data segment.
IOSOPSIDX01	IOS OPS mode index segment.
IOSTS1DAT01	IOS TS1 mode data segment.
IOSTS1IDX01	IOS TS1 mode index segment.
IOSTS2DAT01	IOS TS2 mode data segment.
IOSTS2IDX01	IOS TS2 mode index segment.

4.3 Named Caches

A cache is a block of memory that is used by Sybase to retain and manage data pages that are currently being processed. A *named cache* is a block of memory that is named and used by the RDBMS to store frequently accessed data pages. Assigning a database table to a named cache causes accessed pages to be loaded into memory and retained. The named cache does not need to be allocated to accommodate the entire database table since the RDBMS manages the cache according to use. Named caches greatly increase performance by eliminating the time associated for disk input and output (I/O). There are no named caches that are currently defined for the IOS database. As the DAACs become operational and data access patterns well known, named caches may be defined by ECS development or the maintenance operations staff.

This page intentionally left blank.

5. Database Security

5.1 Approach

The database security discussed within this section is bounded to security implementation within the Sybase SQL Server DBMS. A Sybase general approach to security is adopted as illustrated in Figure 5-1.

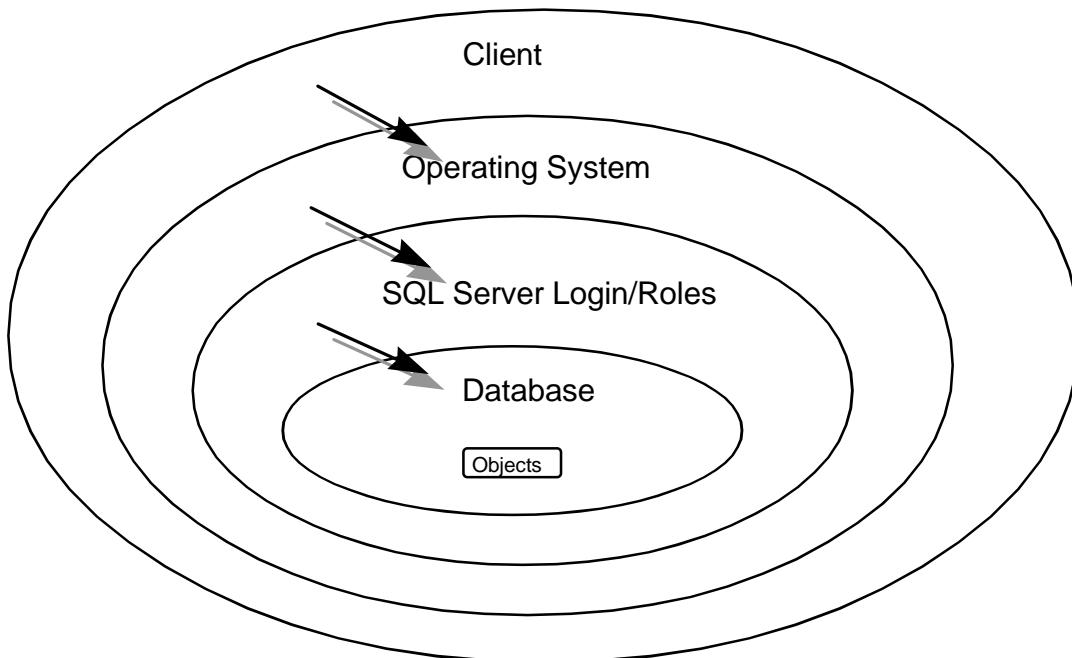


Figure 5-1. Sybase General Approach to SQL Server Security²

5.2 Initial Users

The client (user) requires a SQL Server login to access the DBMS. The login is assigned to a user with certain related permissions for gaining access to particular objects (e.g., database tables, views, commands) within the database. The System Administrator may grant or revoke objects permissions for a login individually or based on defined group or roles.

² Reference Sybase Student Guide: *Advanced SQL Server Administration*.

5.3 Groups

Groups are a means of logically associating users with similar data access needs. Once a group has been defined, object and command permissions can be granted to that group. A user who is member of a group inherits all of the permissions granted to that group. No groups have been initially defined in the IOS Subsystem “default database. The DAACs should define database groups to support the database security requirements of their individual DAACs. Security for local DAAC users should be controlled by assigning each user to the appropriate group.

5.4 Roles

Roles were introduced in Sybase to allow a structured means for granting users the permissions needed to perform standard database administration activities and also provide a means for easily identifying such users. There are six pre-defined roles that may be assigned to a user. A definition of each of these roles follows, as well as a description of the types of activities that may be performed by each role.

System Administrator (*sa_role*): This role is used to grant a specific user permissions needed to perform standard system administrator duties including:

- installing SQL server and specific SQL server modules
- managing the allocation of physical storage
- tuning configuration parameters
- creating databases

Site Security Officer (*sso_role*): This role is used to grant a specific user the permissions needed to maintain SQL server security including:

- adding server logins
- administrating passwords
- managing the audit system
- granting users all roles except the *sa_role*

Operator (*oper_role*): This role is used to grant a specific user the permissions needed to perform standard functions for the database including:

- dumping transactions and databases
- loading transactions and databases

Navigator (*navigator_role*): This role is used to grant a specific user the permissions needed to manage the navigation server.

Replication (*replication_role*): This role is used to grant a specific user the permissions needed to manage the replication server.

Sybase Technical Support (sybase_ts_role): This role is used to grant a specific user the permissions needed to execute *database consistency checker (dbcc)*, a Sybase supplied utility supporting commands that are normally outside of the realm of routine system administrator activities.

The DAACs should review these roles and assign them to the appropriate login and/or groups.

5.5 Login/Group Object Permissions

During initial database installation logins used by the ECS custom code were created and assigned permissions for access to the IOS Subsystem database. In addition, special database installation login, ios_role, was created to support database installation needs. For each login, the level of access is limited to that associated with their login, group or assigned group/role. Object Permissions are set within the installation scripts of the IOS Subsystem for each object and group/role.

Permissions are identified in Table 5-1. A specification of the object permissions is contained in Table 5-2.

Table 5-1. Permission Key

Permission	Description
A	All
S	Select
I	Insert
U	Update
D	Delete
E	Execute

Table 5-2. Group Specifications

Group/Role	SYBASE LOGIN	Permissions Granted					
		A	S	I	U	D	E
None	EcloAdServer	Y					
None	sa	Y					
sa_role	ios_role	Y					

This page intentionally left blank

6. Scripts

The scripts identified in this section may be found in the directory named /ecs/formal/IOS/AdvService/src/Database.

6.1 Installation Scripts

Scripts used to support installation of the IOS database are described in Table 6-1.

Table 6-1. Installation Scripts

Script File	Description
EcloAdDbBuild	Initializes the schema, loads the triggers, initializes the database with data, Initially populates the database, adds server names as user for login, grants permissions to user.
EcloAdDbPatch	Updates a schema to the current database version level.
EcloAdDbDump	Used to dumps specified database prior to commencing with installation. This script is not used to perform regularly scheduled database dumps per DAAC maintenance schedule.
EcloAdDbLoad	Used to load a specified database.
EcDbDesc	List and detail the structure of all database objects in a given database.
EcDbChecksum	Provide row count totals for each of the tables in a given database.

6.2 De-Installation Scripts

Scripts used to support de-installation of the IOS database are described in Table 6-2.

Table 6-2. De-Installation Scripts

Script File	Description
EcloAdDrop	Drops all objects in a specified IOS database.

6.3 Backup and Recovery Scripts

Scripts used to perform backup and recovery of the IOS database are described in Table 6-3.

Table 6-3. Backup and Recovery Scripts

Script File	Description

6.4 Miscellaneous Scripts

Miscellaneous scripts that are applicable to the IOS database are described in Table 6-4.

Table 6-4. Miscellaneous Scripts

Script File	Description
fix_constraints.sql	Adds constraints to tables
grant_iosuser_permission.sql	Grants permissions to users (including software) for access to the IOS database.

Appendix A. Interoperability Entity Relationship Diagram

This page intentionally left blank.

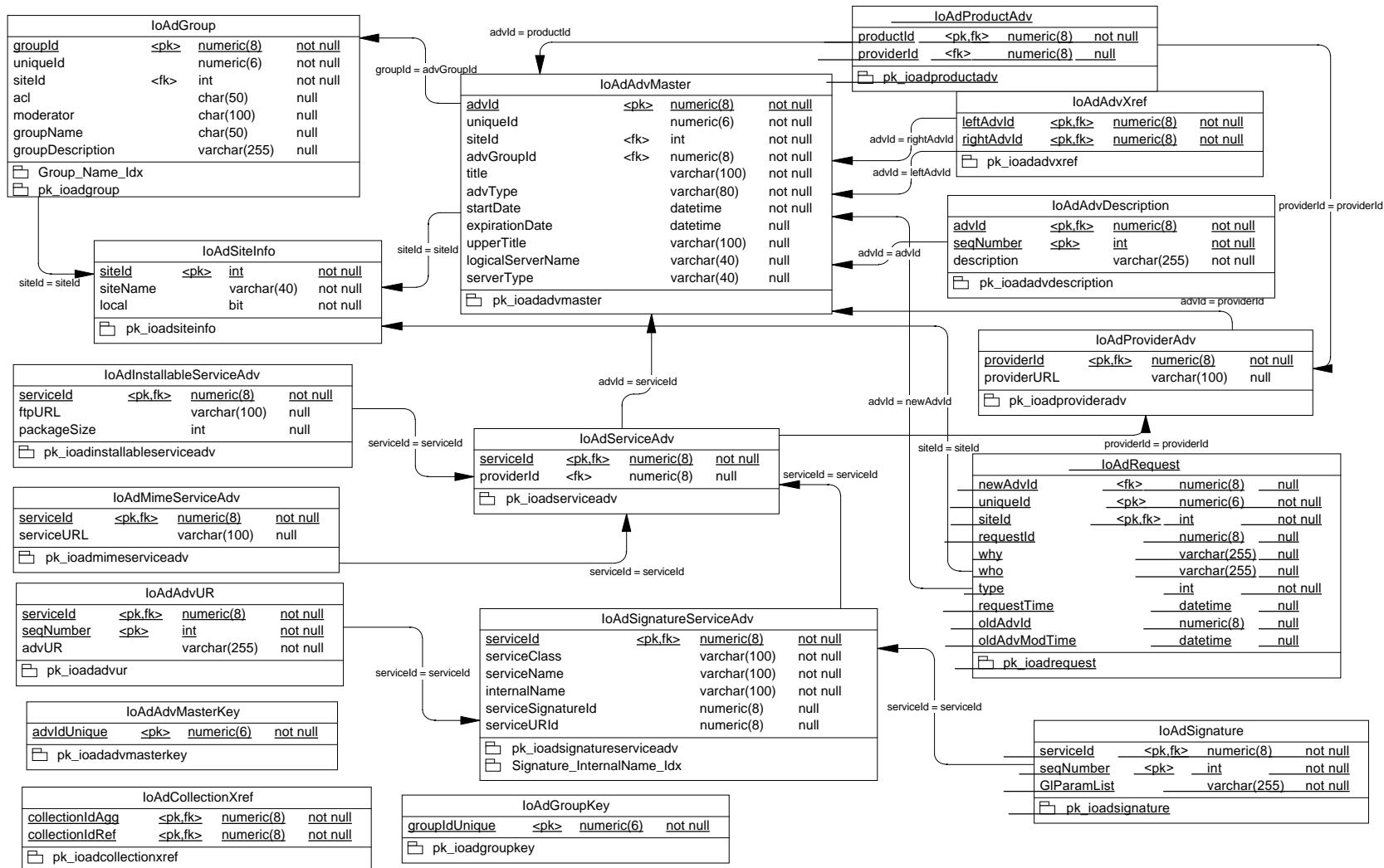


Figure A-1. Advertisements

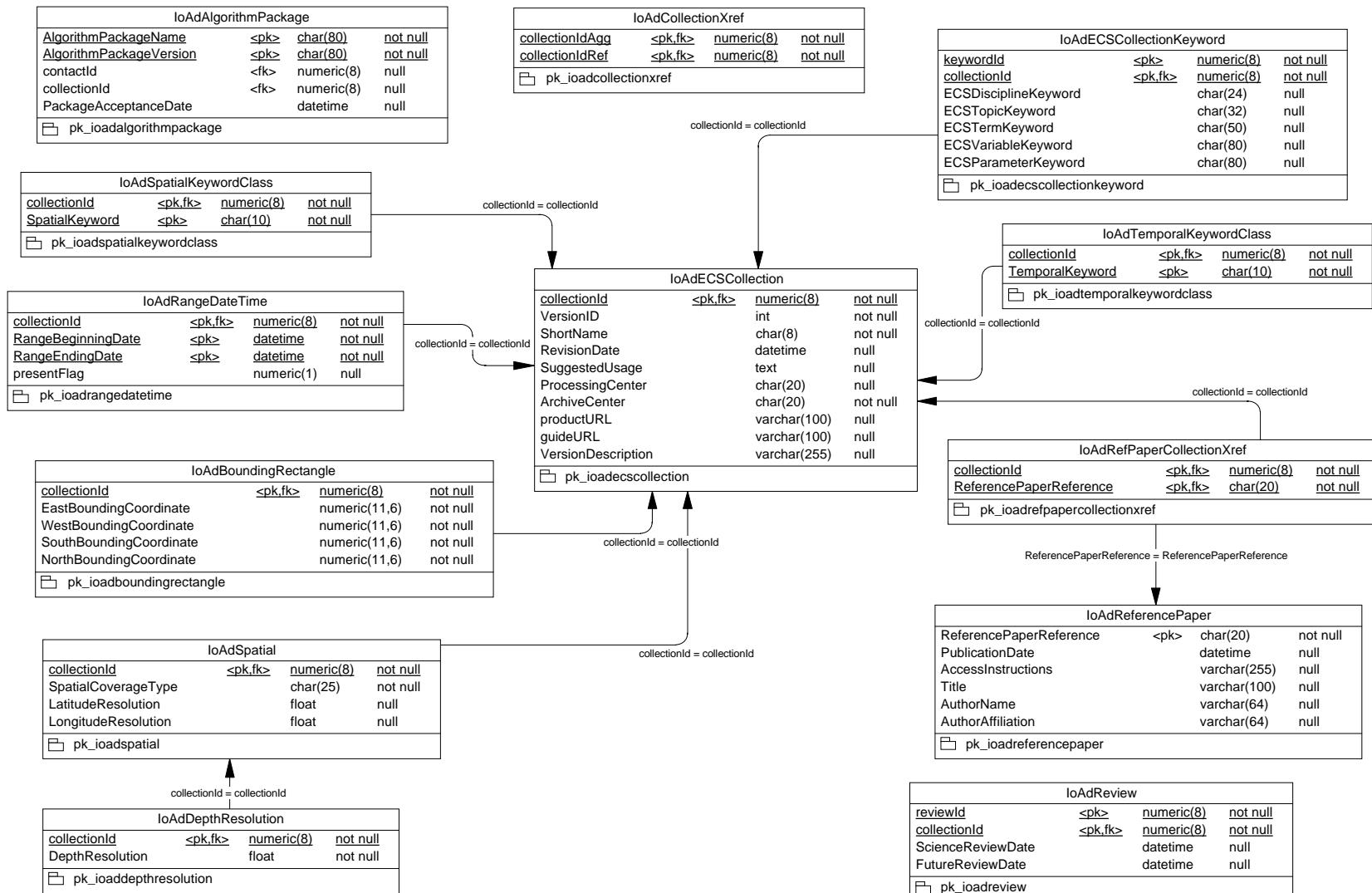


Figure A-2. Collection

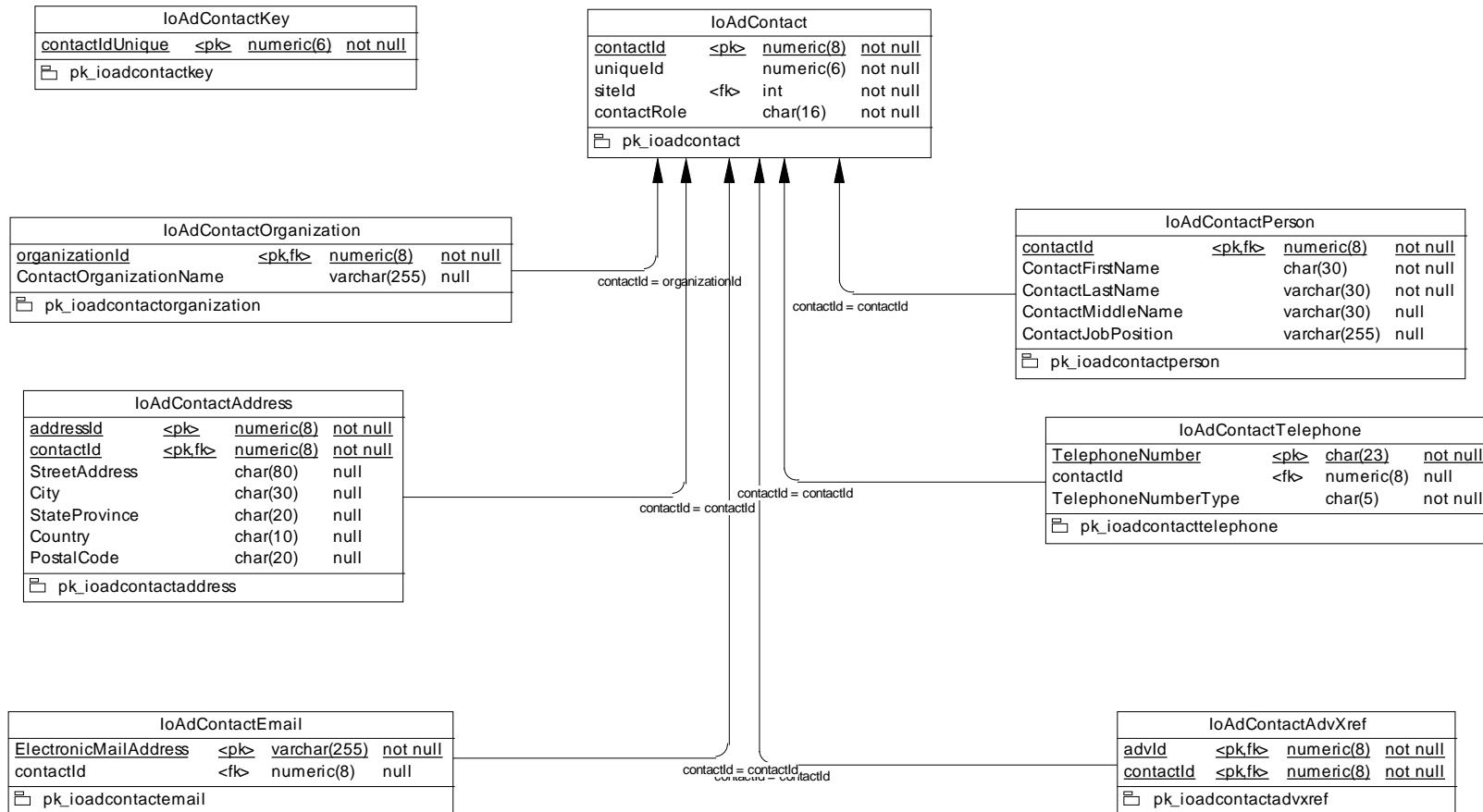


Figure A-3. Collection Contact

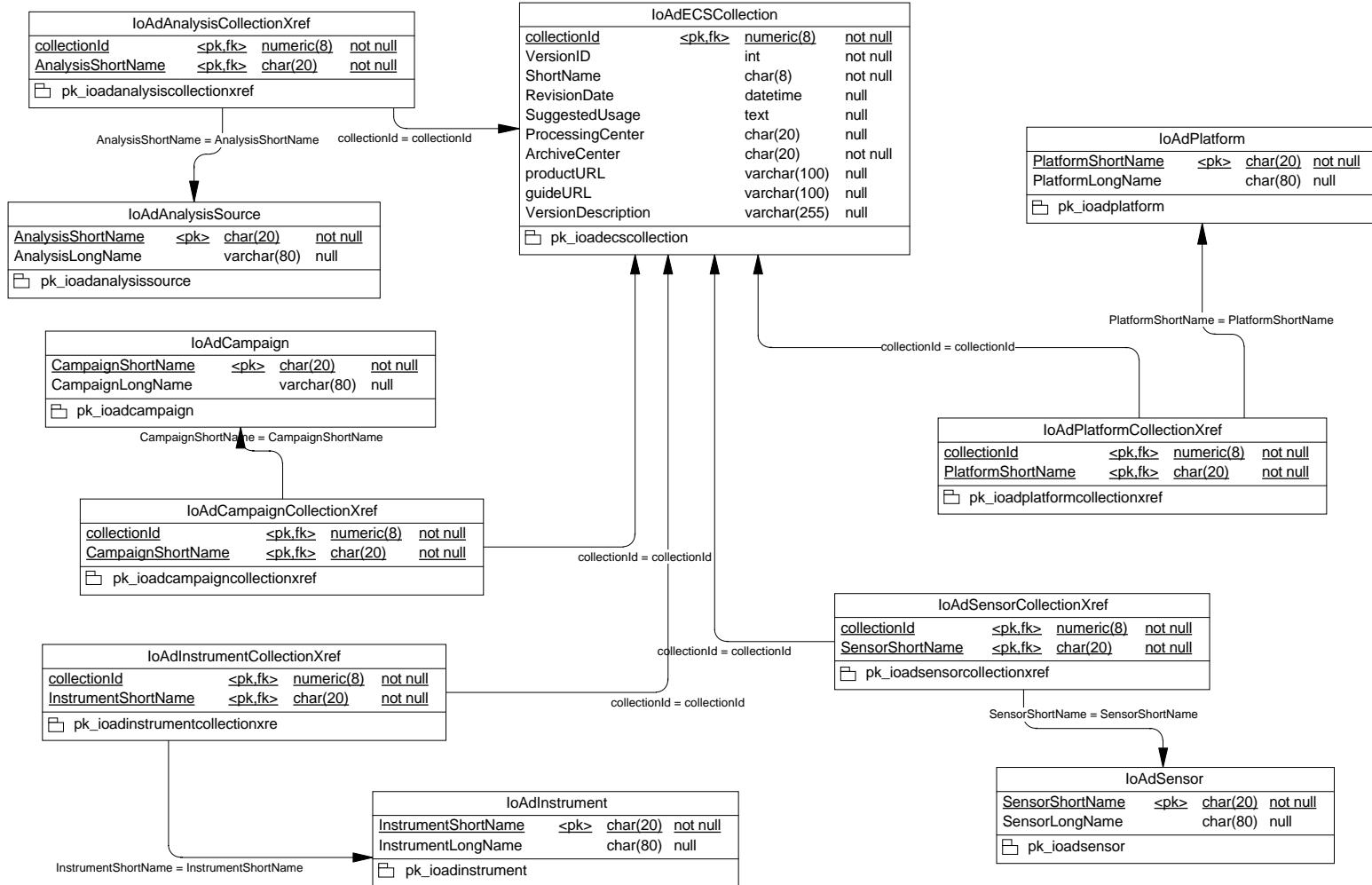


Figure A-4. Data Source

EcDbVersions			
EcDbSchemaVersionID	<u><pk></u>	smallint	not null
EcDbDropVersion		char(64)	not null
EcDbDropDescription		varchar(255)	null
EcDbCurrentVersionFlag		char(10)	not null
EcDbDatabaseName		varchar(255)	null
EcDbDropInstallDate		datetime	null
EcDbSybaseVersion		varchar(255)	null
EcDbSybaseServer		varchar(255)	null
EcDbComments		varchar(255)	null
EcDbUpdateProcess		varchar(255)	null

Figure A-5. Database Versioning

This page intentionally left blank.

Abbreviations and Acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Exchange
CASE	Computer Aided Software Engineering
CD	contractual delivery 213-001
CDRL	contract data requirements list
CI	configuration item
COTS	commercial off-the-shelf (hardware or software)
CSCI	computer software configuration item
DAAC	Distributed Active Archive Center
DBCC	Database Consistency Checker
DBMS	Database Management System
DCN	Document Change Notice
DID	data item description
DMS	Data Management Subsystem
ECS	EOSDIS Core System
EDC	EROS Data Center
EDHS	ECS Data Handling System
EOSDIS	Earth Observing System Data and Information System
EROS	Earth Resources Observation System
ERD	Entity Relationship Diagram
ESDIS	Earth Science Data and Information System (GSFC)
ESDT	Earth science data types
ESN	EOSDIS Science Network (ECS)
FK	Foreign Key
GSFC	Goddard Space Flight Center
GUI	graphic user interface
HDF	hierarchical data format

HDF-EOS	an EOS proposed standard for a specialized HDF data format
HTML	HyperText Markup Language
HTTP	Hypertext Transport Protocol
I/O	input/output
ICD	interface control document
INGST	Ingest Services CSCI
IOS	Interoperability Subsystem
LaRC	Langley Research Center (DAAC)
MSS	Management Support Subsystem
N/A	not applicable
NAS	National Academy of Science
NASA	National Aeronautics and Space Administration
NSIDC	National Snow and Ice Data Center (DAAC)
ODL	Object Definition Language
PCF	Process Control File
PDF	Portable Document Format
PDPS	Planning and Data Processing Subsystem
PGE	Product Generation Executive
PK	Primary Key
QA	Quality Assurance
SDSRV	Science Data Server CSCI
SQL	Structured Query Language
STMGT	Storage Management Software CSCI
SUBSRV	Subscription Server
WWW	World-Wide Web